S I M U L A T I O N

O F A

P A G I N G C O M P U T E R S Y S T E M

A

Thesis

presented to

The Faculty of Graduate Studies and Research

The University of Manitoba

In Partial Fulfillment

of the Requirements of the Degree

Master of Science

in the Department of Computer Science

---

by

William P. Boote

October, 1969

A B S T R A C T

A simulation model of the Manchester University Atlas computer is developed to examine the reliability of simulation results. Job statistics logged by the Atlas computer are used to supply the input job-mix parameters necessary to generate a job stream using Monte Carlo techniques. The characteristics of performance used are the mean duration and the mean relative response of the jobs within various ranges of compute time.

Two different comparisons between the simulation model and a typical Atlas job stream are made. The first comparison involves results gathered under an increasing page demand and the results are extrapolated to the full Atlas paging load. Between 800 and 1000 jobs were processed for each value of the page demand.

The reliability of the simulation results is examined using a page demand function in the second comparison. The mean interarrival time of out-of-core page requests is taken to be a function of the number of in-core pages for each job. Results from the simulation of 1000 jobs are used to compare with the Atlas statistics.

Simulation results obtained by both these techniques are found to be in fairly good agreement with results from logged information.

A C K N O W L E D G E M E N T S

I would like to extend sincere thanks to Professor T. A. Rourke, my thesis supervisor, for his invaluable assistance and guidance during the course of this research.

Secondly, I wish to thank Professors S. R. Clark and A. Wexler for their comments, constructive criticisms and for their time spent in reading this thesis.

I would also like to express thanks to Miss C. McQuarrie for typing this thesis and the operation staff at the University of Manitoba Computer Centre for their co-operation given to me while running the simulation model.

The financial assistance given during the course of this research by the National Research Council of Canada is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

A_{ij}	arrival time of job i of compute range j	14
A_{ij}'	time job i of compute range j selected for assembly	52
A_{ij}^*	time of end of assembly of job i of compute range j	52
A_{ij}''	time job i of compute range j entered on exexution list	52
β_{ij}	blocks transferred by job i of compute range j	13
$\bar{\beta}_j$	mean blocks transferred by jobs of compute range j	11
C_1	paging load proportionality constant	70
C	core storage capacity (pages)	48
γ_i	compute range (γ_i = j) of job i	14
D	drum storage capacity (blocks)	18
d_i	the drum device time used by job i	52
$\bar{d}_j$	mean drum device time used by jobs of compute range j	11
d_r	drum response time	21
$\bar{\delta}_j$	mean rate for out-of-core page requests for jobs of compute range j	14
E	CPU utility (per cent)	55
ε_{ij}	interarrival time for out-of-core page requests of job i of compute range j	21
$\bar{\varepsilon}_j$	mean interarrival time for out-of-core page requests of job within compute range j	21

CHAPTER I

INTRODUCTION

The numerous factors and complexities involved in computer systems do not easily lend themselves to an analytic analysis. As suggested by Penny [1], to be able to obtain accurate results and to evaluate modifications of time-sharing procedures, a simulation study is practically unavoidable.

The analysis of moderately sophisticated computer systems by simulation techniques, although effective [2,3], gives rise to both advantages and disadvantages. The major single advantage is that simulation often provides the only feasible approach to the analysis of computer systems, which in general resist analytic methods. Disadvantages include the time, cost and difficulty in the development of an appropriate model and the actual computer time required to obtain reliable results [7].

In order to be able to define the object system both analytic and simulation techniques generally involve assumptions and approximations of variables, parameters and of meaningful analytic relationships which are not well behaved mathematically. Analytic studies usually involve the properties of the populations of the input and output variables, and the results are therefore applicable to an infinite number of jobs.

Although simulation methods usually require fewer approximations than analytic methods, simulation studies in which the desired job-stream is defined by Monte Carlo techniques result in output variables dependent on the number of jobs processed during each simulation run. This indeterminateness of the simulation results is referred to as the reproducibility.

Fishman [4] considered the problem of the reproducibility of simulation results in relation to the number of jobs processed during each simulation run. He gives recognition to the degree of autocorrelation present in the output statistics which are in turn averaged to give the results of a simulation. A body of empirical data from which it is possible to calculate the reproducibility for specific situations has been studied by Wren [6]. However, the problem of determining the reproducibility of a set of results by means other than repetitive calculations is by no means solved [5].

Assuming that the reproducibility of a set of simulation results can be relied upon, a further equally important problem arises. This problem, referred to as the reliability, concerns the discrepancy between the simulation results and results logged for a real computer system with the same design characteristics. This same problem applies equally to analytic studies.

In most computer system models which set out to simulate the processing of a large number of jobs, it is assumed that events which either occur with an interarrival time of less than about a milli-

second, or last less than about a millisecond, do not affect the overall operation of the computer system and may therefore be ignored. For example, the request by a program to use a magnetic tape deck may give rise to many elementary[1] circuit operations but simulation models usually take account only of the total length of time of the transfer [2,3].

This type of approximation, plus others which are described in later chapters, tend to place a certain amount of doubt on the reliability of simulation and analytic results. Very few direct comparisons between a simulation model and a real system have been carried out.

In most computer-system models which set out to simulate the processing of a large number of jobs, it is assumed that the processing of jobs is adequately represented as a series of demands for CPU attention interspersed with demands on the other facilities of the computer system. Input-output operations are simulated as elementary transmissions of data having a prescribed length. Such approximations are rather gross considering the number of operations carried out in a real system. It is therefore necessary to establish that all the features of the real system which are essential to its accurate simulation have been retained in any particular simulation model.

---

1. elementary in the sense that they are not broken down.

Scherr has carried out a comparison between a next-event type of simulation model (which produced jobs by Monte-Carlo techniques) and monitored results from the Compatible Time Sharing System (CTSS) at Mass. Inst. of Technology [2]. The CTSS is an interactive system with approximately 250 users. The CPU, drum and disc operate serially and are not overlapped. Scherr has succeeded in showing that a model which includes the gross approximations discussed above is capable of modelling the CTSS successfully. This is not to say that all attempted simulations of this type will be equally successful, but rather the important conclusion is that approximations of this level can produce accurate results.

In addition to this comparison Scherr carried out calculations using a Markov analytical model, which necessarily involved even more gross approximations such as the neglect of the hardware and job-stream details included in the former model, and he again found the comparison with the real system to be favourable. This too is an important conclusion since a rather large number of parameters were used to describe the job-stream in the former model, and such a large number of parameters would make the study of hypothetical computer systems, where the job-stream cannot be projected in such detail, extremely tedious.

Work has been carried out on hypothetical computer systems using a next-event type of simulation model [6,8,9] and further work on systems with autonomous CPU, drum and discs or tapes is planned

with an extended version of the same model. This model differs from Scherr's in three important respects:

(i) hardware operations are not treated in as much detail (eg. the positioning of the drum read heads relative to the information to be read is not considered);

(ii) fewer input parameters are used to describe the job-stream and its demands on the system (eg. different stages such as compilation or execution are not distinguished); and

(iii) the operation of the CPU drum and tapes are autonomous.

Although the work carried out by Scherr goes a long way towards justifying this slightly different level of approximation, it is important in view of the projected work to try to establish the validity of the model.

The purpose of this thesis is to establish that the above simulation model is capable of simulating a real system successfully by incorporating any additional essential features of the real system into the model. The assumption thereby made is that if the simulation model (appropriately modified) is capable of simulating a real system successfully, then the simulation results corresponding to systems of simpler or equal complexity to the real system should be equally well simulated.

Statistics on the operation of the Atlas computer at Manchester University are readily available [10] and the information is such that

the simulation of this system by modifying the parent simulation model is not too difficult a task, and the system is sufficiently complicated to allow plenty of scope for the simulation of simpler systems. Accordingly, this system was selected as the real system for comparison with the simulation model modified to include paging and other features of the Atlas computer. As some variations are found between the various Atlas computers, the reference to the Atlas system or computer throughout this research is specifically the University of Manchester Atlas computer.

The concept of multiprogramming in a paging computer such as the Atlas computer is initially easily understood. The details are in fact complex. In paging systems all information addressable by the central processor is structured into manageable segments called pages. In a similar manner the available drum and core storage is sub-divided into blocks or pages such that any block of a program may be located or relocated on a block of drum or core storage. At any one moment during execution the blocks comprising a particular job may therefore be distributed between both drum and core storage. If only the blocks which are currently in core storage are required then execution may proceed. In general execution proceeds until information is required which is not found in core storage. The required block of information is fetched on a demand basis.

In systems which allow more than one program to share core storage it is usually the case that core storage is full as a page is required

from drum storage. In such an event, a page is chosen to be transferred back to drum storage. The mechanism making this page selection is termed the replacement strategy.

In the development of the model for comparison with the Atlas computer, the extensions made to an existing simulation model included the simulation of such operations as the assembly of jobs (that is, gathering all information necessary to execute a job), the loading of a job into core storage, the system output and user magnetic tape transfers, the removal of jobs from the system and the scheduling of both jobs and the central processor. The core storage has a finite capacity and a maximum of two jobs are allowed to share storage with another job maintained in the assembly phase if possible. Jobs awaiting assembly are held in the assembly queue.

As part of the supervisory system of the Atlas computer, two algorithms are of primary interest, the job-scheduling algorithm and the replacement algorithm or drum-learning program. The job-scheduling algorithm determines which job is selected from the assembly queue next on the basis of information supplied by user requests and the type of jobs required in accordance with the Atlas scheduling philosophy.

The drum-learning program is that part of the supervisor which contains the replacement strategy for the decision as to which page is to be removed from core storage if core is full and yet another page is to be fetched.

CHAPTER II

ATLAS SYSTEM AND SIMULATION MODEL

2.1 INTRODUCTION

The nucleus of the simulation model developed for this study was first used by Chai [8], and later by McDonald [9] and Wren [6] for the simulation of hypothetical time-shared computer systems. This nucleus was further extended to include operations which are particularly characteristic of the University of Manchester Atlas computer. The salient features which characterize the Atlas computer are well publicised in the literature. Thus, the description of the Atlas computer at Manchester University, which is necessarily included with the description of the simulation model, is only in order to make approximations involved in the simulation model apparent. A more complete description may be found elsewhere [10,11,12].

As in the Atlas configuration, the model has a finite capacity of physical storage of 165 units or blocks, of which 32 units are main core storage and the remaining 133 units consist of magnetic drum storage. In addition to the 5 magnetic tapes available to the user, the Atlas computer uses 3 system magnetic tapes, of these, only the system output tape is simulated by the model. The system input and compile magnetic tapes are ignored. Autonomous magnetic

tape and drum transfers allow job execution to be overlapped while these operations are in process. The fundamental operation of the simulation model is to allocate the central processor during drum and magnetic tape transfers between a maximum of two jobs on the execution list and supervisory tasks. Supervisor overheads of magnitude less than 1 millisecond are omitted in the simulation model.

The simulation model may be conveniently described in four major sections:

(1) The Job Stream: which describes properties of the Atlas job-stream, the input job-mix parameters to the simulation model, the job generation techniques used and the resulting job-stream;

(2) Operation of Atlas System and Model: which outlines the main features of the Atlas supervisor and describes the simulation model and its operation;

(3) Paging in Atlas and Simulation Model: which describes the Atlas paging operations, replacement strategy and the simulated counterpart for these in the model; and

(4) Performance Characteristics: which discusses the characteristics of performance of the Atlas system and simulation model.

2.2 THE JOB-STREAM

This section describes briefly the observed Atlas job-stream statistics collected especially for this study. These describe the

job-stream observed at the University of Manchester during a typical week of operation. Secondly, this section describes the job generator, the individual jobs generated and the resultant job-stream which represents the work load for the simulation model used to simulate the Atlas computer.

2.2.1 ATLAS JOB-STREAM

Properties of the user's job population are supplied by the Atlas statistics. Each job processed by the Atlas computer is assigned to one of five classes or categories, depending in which compute range $[l_j, u_j]$ the requested compute time, ξ_{ij}, for job i belongs. The boundaries l_j and u_j are partly fundamental to the Atlas scheduling in defining each job to be one of: a short job (range $[l_j, u_j]$, j = 1, 2, and 3) or a long job (range $[l_j, u_j]$ j = 4, 5). However, all the compute ranges chosen are in accordance with those used in a discussion of the performance of the Atlas computer [10]. Thus jobs are classified in one of several compute ranges bounded above and below by u_j and l_j respectively. Any job requiring at least one user magnetic tape is referred to as a tape job.

The Atlas statistics [13] used in this study were determined from information logged by the Atlas computer as it completes processing each job. These statistics contain information concerning the:

(a) compute time;

(b) requested compile store;

(c) requested execution store;

(d) number of user magnetic tapes;

(e) printer output lines;

(f) blocks transferred;

(g) drum device (one level store) time; and

(h) observed duration.

This basic information was used to obtain the input job-mix parameters described below. Table 2.1 shows an example of the statistics gathered for these parameters collected during a typical week of operation:

(a) the boundaries of the computer range, $[l_j, u_j]$;

(b) the mean central processor or compute time requested, denoted by $\overline{\xi}_j$;

(c) the mean storage blocks requested, denoted by $\overline{R}_j$;

(d) the mean number of magnetic tapes requested denoted by $\overline{U}_j$;

(e) the mean number of printer output lines observed denoted by $\overline{nl}_j$;

(f) the mean number of block transfers observed, to or from the magnetic tapes, denoted by $\overline{\beta}_j$; and

(g) the mean one-level store or drum device time observed, denoted by $\overline{d}_j$.

T A B L E 2 . 1

EXAMPLE OF ATLAS STATISTICS

	COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	COMPILE STORE (BLKS)	EXECUTION STORE (BLKS)	NO. OF TAPES	PRINTER OUTPUT (LINES)	BLOCKS TRANS. (BLKS)	ONE-LEVEL STORE (SEC)
j=1	<1	0.3	360.0	61.9	34.7	0.8	22.7	423.6	1.0
j=2	1 - 8	3.8	864.0	65.3	37.3	0.4	132.5	344.2	6.3
j=3	8 - 120	33.1	1133.0	77.4	55.5	0.5	732.1	449.6	21.6
j=4	120 - 960	284.0	194.0	75.5	56.0	0.5	1132.4	452.5	54.6
j=5	>960	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Certain difficulties were encountered in the calculation of the mean interarrival times from the logged information, and as a result the interarrival times were calculated from the number of jobs in each compute range. Several simplifying approximations concerning the input statistics were adopted.

The mean number of blocks transferred, $\overline{\beta}_j$, was calculated for the Atlas statistics as:

$$\overline{\beta}_j = \sum_{i=1}^{n_j} \beta_{ij}/n_j \qquad \text{.....(2.2.1)}$$

where n_j represents the number of jobs processed for compute range j. The time necessary to transfer β_{ij} blocks (without rewinds and searches) is equivalent to the rewind time, the search time and the block transfer time for job i of compute range j. In effect the rewind and search time is converted to an equivalent number of block transfers. Assuming that a tape is rewound or searched a number of times equal to the number of block transfers, the average time taken by a tape for the completion of a single request is approximately the time necessary to transfer 3 to 7 blocks without rewind or searches [10]. Tape requests are simulated as transfers of groups of 8 blocks, slightly in excess of these approximate figures.

Secondly, no distinction is made by the simulation model between compilation and execution. Thus the mean storage blocks requested, $\overline{R}_j$, is taken to be the mean value of the compile and execution store given in the Atlas statistics for each compute range.

Lastly, it is convenient to assume that the compute times and interarrival times are exponentially distributed and not dependent upon time. These are fairly standard assumptions [1,2,3].

2.2.2 SIMULATED JOB-STREAM

A series of n entities or jobs are produced by the job generator, using the described job-mix parameters, operational parameters, Monte Carlo techniques, normal and Poisson generating functions [14]. These jobs compete to be processed by the central processor. It follows then that the generated job descriptions directly determine the scheduling of each job through the simulation model. For each job i in the job-stream, the eight job parameters which completely define a job of compute range j are:

(1) the compute range or category number, denoted by $\gamma_i (\gamma_i = j)$;

(2) the arrival time of each job, denoted by A_{ij};

(3) the compute time or central processor time requested, denoted by ξ_{ij};

(4) the number of units or blocks of physical storage requested, denoted by R_{ij};

(5) the number of user magnetic tapes requested, denoted by U_{ij};

(6) the number of output blocks requested, denoted by Θ_{ij};

(7) the mean rate of out-of-core page requests, denoted by $\overline{\delta}_j$; and

(8) the mean rate of block transfer (to user tapes) requests, denoted by $\overline{\phi}_{jm}$, $(m = U_{ij})$.

A compute range number is assigned to each job generated. This number, denoted by γ_i identifies one of the L compute ranges to which each job must belong. To clarify, if a job i has an assigned compute range number γ_i equal to j, the central processor time requested for that job lies in the range j. Jobs requesting the least compute time have a compute range number equal to 1, while those requesting the greatest central processor time would have γ_i equal to L.

The arrival time, A_{ij}, of a job i in the job-stream belonging to compute range j is the accumulated value of the interarrival times λ_{ij} of range j. Explicitly, if the interarrival time for job i of compute range j is λ_{ij}, the arrival time for that job is:

$$A_{ij} = A_{i-1,j} + \lambda_{ij} \quad . \qquad .....(2.2.2)$$

The interarrival times are assumed to be exponentially distributed around the mean $\overline{\lambda}_j$, which is determined by:

$$\overline{\lambda}_j = T/n_j \quad , \qquad .....(2.2.3)$$

where T is the time over which the Atlas statistics are observed and n_j the number of jobs observed in compute range j.

The job generator uses the input job-mix parameters describing the job-stream to be simulated over a specific period of time, T. The system efficiency or central processor utilization of the Atlas system at Manchester is 61%, as stated in a previous publication [10]. The duration over which each set of Atlas statistics are gathered is estimated by:

$$T = \sum_{j=1}^{L} n_j \bar{\xi}_j / 0.61 \quad , \qquad \text{.....(2.2.4)}$$

where n_j and $\bar{\xi}_j$ are input job-mix parameters for each compute range j, denoting the number of jobs processed and the mean requested compute time for the n_j jobs observed in each of the L compute ranges.

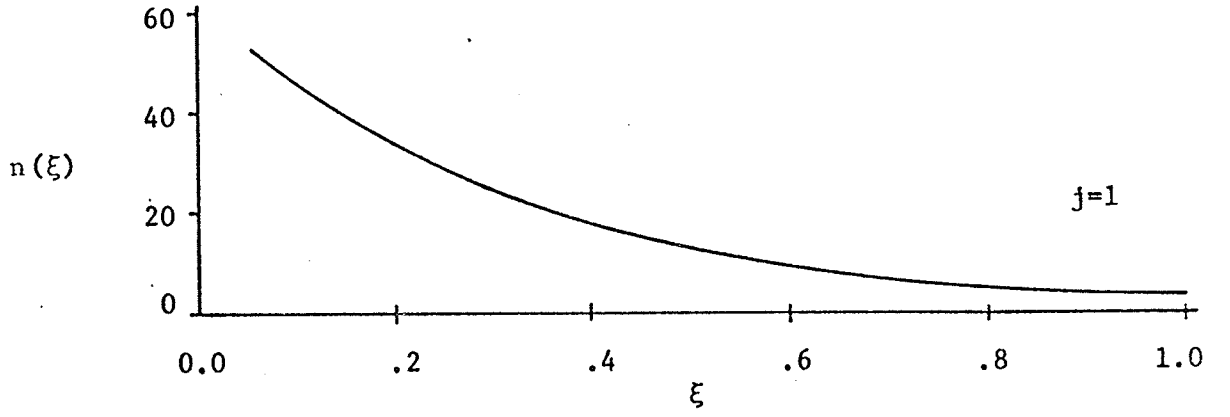
The compute time or central processor time requested, ξ_{ij}, for each job i of compute range j in the job-stream is generated such that the variate $\xi_{ij} - 1_j$ comes from an exponential distribution with a mean $\bar{\xi}_j - 1_j$ and which is truncated at $u_j - 1_j$. The mean compute time for compute range j is $\bar{\xi}_j$, where 1_j and u_j are the lower and upper limits. Because the exponential distribution is truncated at $u_j - 1_j$, the observed mean of the ξ_{ij} variables generated in this manner is not $\bar{\xi}_j$ but rather the mean over the range $[1_j, u_j]$ given by:

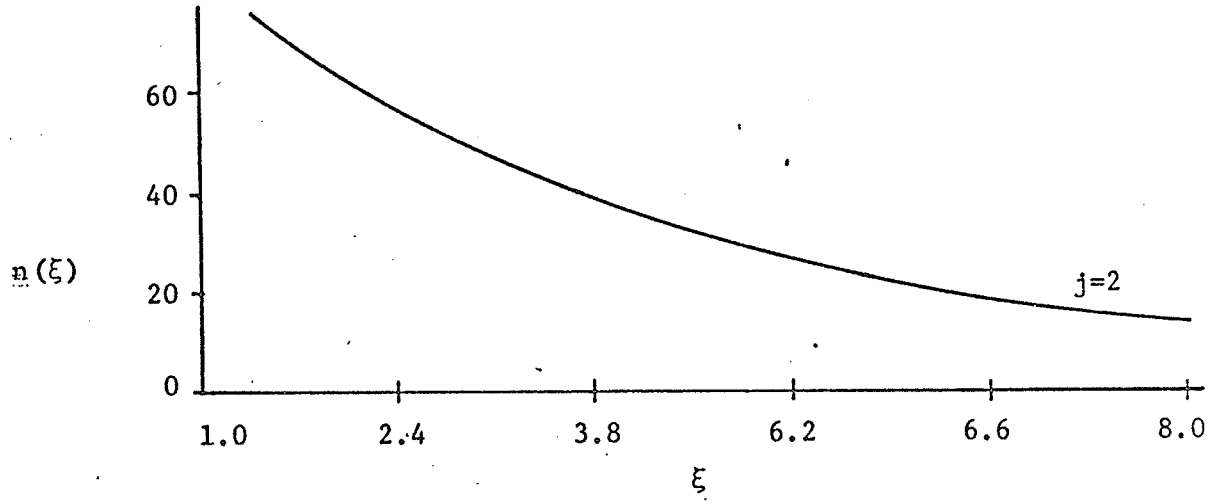
$$\int_{1_j}^{u_j} \frac{\xi_{ij}}{\bar{\xi}_j - 1_j} e^{-(\xi_{ij} - 1_j)/(\bar{\xi}_j - 1_j)} \, d\xi_{ij} . \qquad \text{.....(2.2.5)}$$

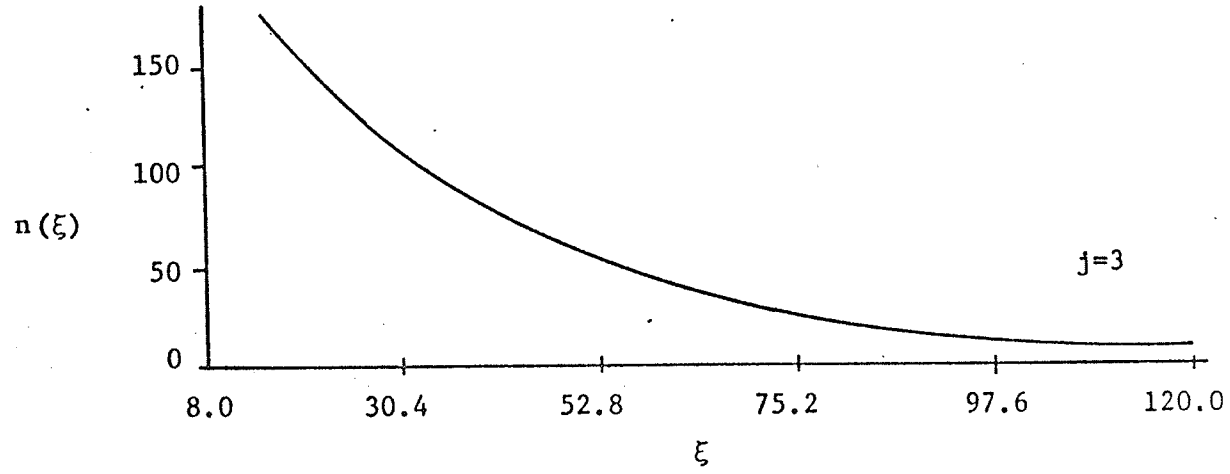
From the Atlas statistics shown in Table 2.1, it was determined (for the compute range with the greatest discrepancy j=2), that for the $\xi_{ij} - 1_j$ variables greater than 2.5 times the mean $\bar{\xi}_j - 1_j$, the discrepancy is less than 8.3%. Figure 2.1 shows the frequency distribution for the n_j jobs in each compute range with mean of approximately $\bar{\xi}_j$.

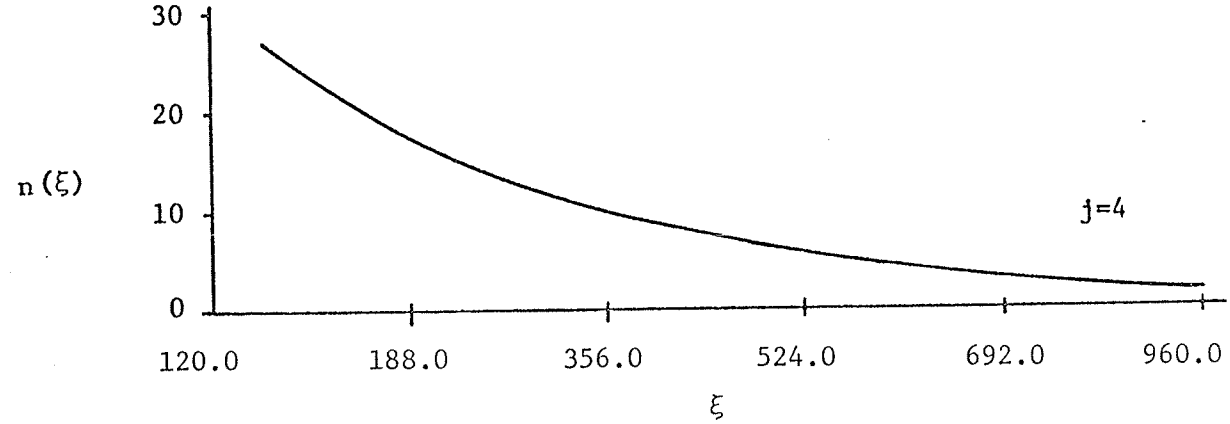
The number of units or blocks of physical storage requested, R_{ij}, for job i is taken to be normally distributed about the mean requested

FIGURE 2.1 FREQUENCY DISTRIBUTION OF $n(\xi)$ VERSUS ξ









storage, $\overline{R}_j$, with standard deviation of $\overline{R}_j/4$. Using the Atlas statistics, the mean $\overline{R}_j$ was calculated by taking an average of the mean compile and execution storage requests. A minor restriction was imposed on the upper limit for the amount of storage that may be requested is taken to be 133 units, which represents the amount of drum storage, D, available. Since each job is assumed to require a certain amount of storage, the storage requested, R_{ij}, for each job is a positive non-zero integer such that $1 \leq R_{ij} \leq 133$.

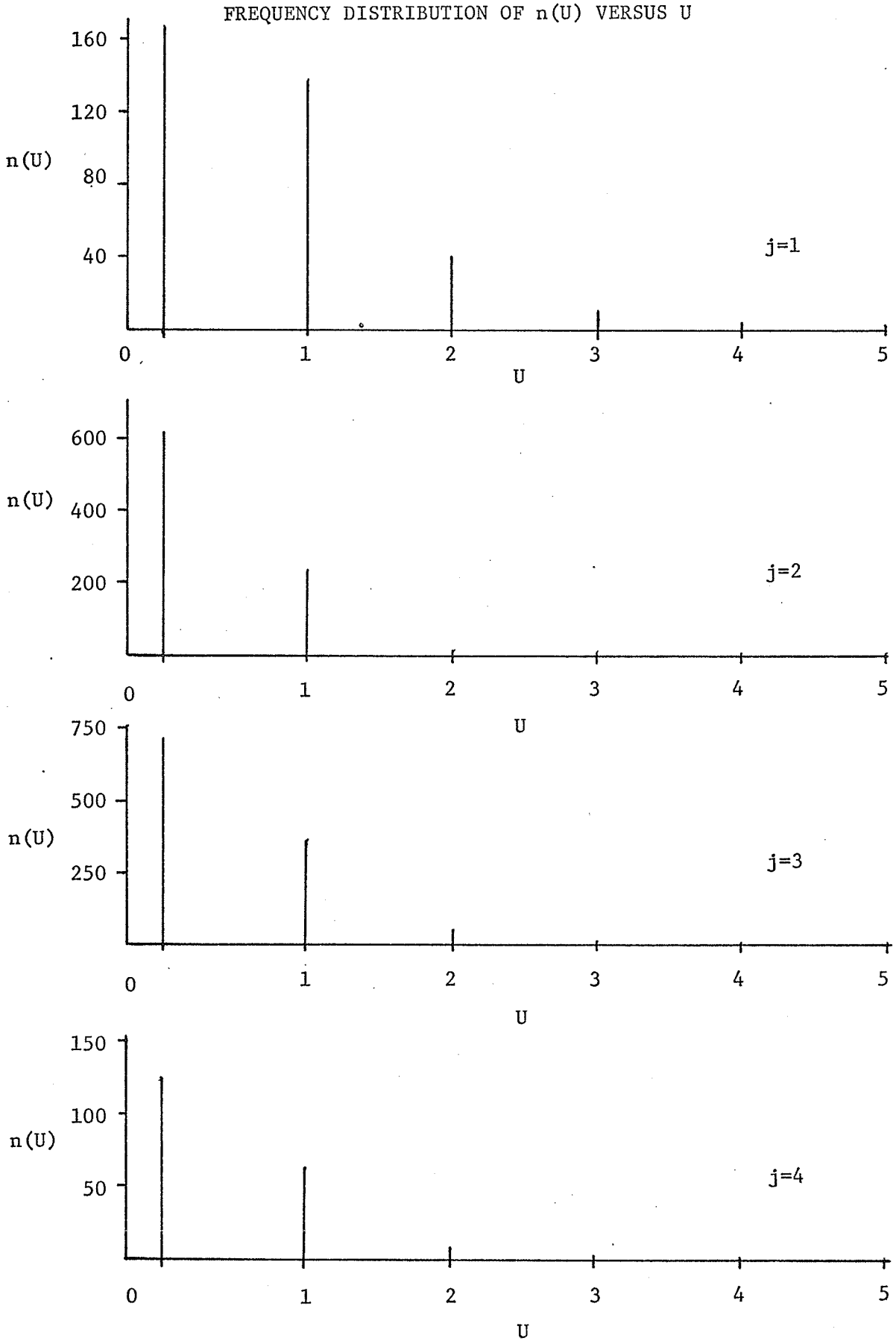
The number of user magnetic tapes requested, U_{ij}, by the i^{th} job of compute range j is geometrically distributed around the mean $\overline{U}_j$. The variate U_{ij}^* is exponentially distributed about the mean $\overline{U}_j$ and U_{ij} is calculated by rounding U_{ij}^* to the nearest integer. Any value not in the range [0, mt], where mt denotes the maximum number of available user tapes, are discarded. In the Atlas system mt equals 5. Because the exponential distribution is truncated at mt + 0.5 and the variates U_{ij}^* rounded to integers, the mean of the U_{ij} generated variables is not $\overline{U}_j$ but rather the mean of a geometric distribution given by:

$$\frac{e^{-0.5/\overline{U}_j}}{1 - e^{-1/\overline{U}_j}} \quad . \qquad \text{.....(2.2.6)}$$

Again, using the Atlas statistics for this study (Table 2.1) the greatest discrepancy between $\overline{U}_j$ and the mean of the geometric distribution (j = 2) is less than 23%. Figure 2.2 shows the frequency distribution for the n_j jobs in each compute range j over the range of user magnetic tapes [o, mt], distributed about mean $\overline{U}_j$.

FIGURE 2.2

FREQUENCY DISTRIBUTION OF n(U) VERSUS U



The number of output blocks requested, Θ_{ij}, is normally distributed about the mean $\overline{\Theta}_j$ with a standard deviation of $\overline{\Theta}_j/4$. The mean number of output blocks transferred to the system output or dump tape, $\overline{\Theta}_j$, is considered to be a function of the mean number of printer output lines, $\overline{nl}_j$, observed for each compute range j. The basic unit for output is taken to be blocks of output. This is reasonable, since all output is first buffered and then transferred to the system output tape a block at a time. The mean number of output blocks is related to the mean number of printer output lines by the equation:

$$\overline{\Theta}_j = \overline{nl}_j/16 \quad , \qquad \text{.....(2.2.7)}$$

where the number of printer output lines per output block is taken to be 16. Assuming that each job has at least some output, this approximation used at the University of Manchester is not unreasonable. In the simulation model, the minimum number of blocks of output is taken to 1.

When a job has received its full compute time, the output blocks are transferred in one long tape transfer. Again, this represents an approximation to the Atlas system, since output may occur during execution in addition to occurring automatically at the end of each job.

The mean rate of out-of-core page requests, $\overline{\delta}_j$, for all jobs belonging to compute range j represents the mean rate at which a job demands that a page be transferred from drum into core storage. Because of the limited storage available, any required page may or may not be in core storage. If not currently in core storage, the required page

is transferred from drum into an empty page of core storage. In order to always maintain an empty page for drum to core page swapping, another page may be required to be transferred out to drum storage. As the drum becomes available upon completion of a drum-to-core storage transfer, the latter transfer is initiated if necessary.

During the interval of execution between each successive request for an out-of-core page, pages resident in core storage may be accessed in some sequence. As stated [10], the out-of-core page requests represent only 0.01% of the total number of pages accessed. Because of the length of time taken for each simulation run, generating all page requests was found to be impractical. For this reason, only out-of-core page requests are generated by the simulation model.

The interarrival time for out-of-core page requests, ε_{ij} is exponentially distributed about the mean interarrival time $\overline{\varepsilon}_j$ for all jobs in compute range j and describes the mean length of the execution interval between successive out-of-core page requests.

The mean number of out-of-core page requests, $\overline{\eta}_j$ for all jobs belonging to compute range j is defined as:

$$\overline{\eta}_j = f\ \overline{d}_j/d_r \quad , \qquad \text{.....(2.2.8)}$$

where $\overline{d}_j$ is the mean drum device time used for swapping pages both into and out of core storage and d_r is the drum response time. The constant, f, is defined as a proportionality constant, such that $f\ \overline{d}_j$ represents the mean drum device time used for swapping pages from drum to core storage. The mean paging rate, $\overline{\delta}_j$, for all jobs of compute range j may be expressed as:

$$\overline{\delta}_j = \overline{\xi}_j / \overline{\eta}_j \quad . \qquad \text{.....(2.2.9)}$$

The mean block transfer request rate, $\overline{\phi}_{jm}$, represents the number of block transfer requests to or from magnetic tape per unit of processing time. For a job belonging to compute range j, requesting m magnetic tapes, the mean rate is expressed as $\overline{\phi}_{jm}$. The mean number of blocks transferred, $\overline{\beta}_j$, averaged over all the n_j jobs within compute range j, is assumed to be directly proportional to the number of magnetic tapes U_{ij} requested by each job in compute range j. The mean block transfer request rate, $\overline{\phi}_{jm}$, (where $m = U_{ij}$) may be expressed as:

$$\overline{\phi}_{jm} = \overline{\beta}_j g(m) / \overline{\xi}_j \quad . \qquad \text{.....(2.2.10)}$$

The function g(m) represents an arbitrarily chosen function in terms of m the number of magnetic tapes requested and P_{jk} the probability of a job requesting k tapes, such that:

$$g(m) = m / \sum_{k=0}^{mt} k P_{jk} \quad . \qquad \text{.....(2.2.11)}$$

The interarrival time between block transfer requests, Ω_{jm}, is assumed to be exponentially distributed about the mean interarrival time, $\overline{\Omega}_{jm}$, ($\overline{\Omega}_{jm} = 1/\overline{\phi}_{jm}$), for each job belonging to compute time range j requesting m magnetic tapes.

The job generator uses normal and Poisson generation functions to generate the eight defined parameters which completely define each job in the job-stream. A generation list maintained by the job generator consists of one slot for each compute range. When there is a vacancy, another job is generated for the appropriate compute range.

When the time of arrival, A_{ij}, of any job on the generation list is equal to the simulated time, that job is transferred from generation list to the job list or assembly queue, where it awaits selection for assembly. Alternatively, if the assembly queue is full, the generated job remains on the generation list until a vacancy appears.

2.3 OPERATION OF ATLAS SYSTEM AND THE MODEL

This section describes the necessary details of the Atlas supervisor in order that the logic of the simulation model becomes apparent. A more detailed description concerning the Atlas supervisor, scheduler, dynamic storage allocation and storage organization have previously been presented [12,11,15,17]. Secondly this section presents in detail the structure and operation of the simulation model used in this study.

2.3.1 OUTLINE OF ATLAS OPERATING SYSTEM

The Atlas operating system is designed to achieve the maximum of overlapping of system functions which may proceed concurrently [16]. For example, execution may be overlapped with magnetic tape transfers, drum transfers, and peripheral input-output transfers. The Atlas supervisor [12] consists mainly of five logically distinct parts, the input supervisor, job scheduler, central executive, magnetic tape supervisor and output routine. The supervisor controls all those system functions which are

a result of the execution of a job appearing on the execution list. During the processing of a job, the various parts of the supervisor communicate, so as to maintain the fullest activity of the computing system.

As a job is presented to the computer system, the input supervisor makes a job entry for each job into the job list or assembly queue. This job entry consists of a description of the system resources requested by that job and is used by the job scheduler. The object of the job scheduler is to maintain a supply of jobs on the execution list and arrange as far as possible to have in storage a magnetic tape job, a non-tape job, and a job in the assembly phase. During the assembly phase, a job is prepared so that upon entry on the execution list, execution may begin without any delay. This requires that job scheduler is able to select the necessary job required from the jobs available. It is suggested [11] that efficient scheduling of jobs through the system increases the demand on the computing system, thereby increasing the system utilization.

As a job enters the system, it is streamed into one of three streams according to the information supplied by the job description. Briefly, the three streams consist of a stream for all tape jobs, a stream for all short jobs and a stream for all long jobs. As both short and long jobs do not request any user magnetic tapes, a long job is defined to be any job requesting an excess of two minutes compute time.

One of the tasks of the scheduler is to assemble in advance jobs that may begin compilation and execution immediately upon entry onto the execution list. Thus, when a particular type of job is required, the scheduler is activated, consults the appropriate stream and searches for the next available job to be assembled.

Assembly of a job involves the collecting of the information required before execution of the job may begin. For example, a completely assembled job has all magnetic tapes mounted and storage allocated in combined core and drum storage. When a vacancy appears, the scheduler again is activated, the job is then entered onto the execution list and the priority of the job established. The priorities in order consist of jobs assigned top priority by operator, magnetic tape jobs, non tape jobs and jobs assigned lowest priority by the operator. In the simulation model, the first and last priorities are ignored and magnetic tape jobs are regarded as highest priority.

The central executive is that part of the supervisor responsible for the monitoring of all executing jobs. Secondly, this part of the supervisor allocates to each job a block directory, which defines the relative address and physical position of each block in storage. During execution, blocks belonging to a particular job may or may not be in core storage which consists of 32 directly accessible pages. As a particular page in core is referenced, this page is located by hardware functions [16]. However, when a block is required which is not currently in core storage, the central executive is entered to arrange the transfer

of the required block from drum into an empty page of core storage. For the duration of this operation the job requesting the transfer is halted by the supervisor. If core storage is full, the central executive selects by means of a drum-learning program, a page to be transferred to drum storage. Thus, an empty page is always maintained in core storage.

The drum-learning program described in detail elsewhere [17] predicts the page in core storage, which is not expected to be required for the largest period of time.

As the paging-in operation is complete, the drum transfer routine is again entered and the paging-out operation initiated. A queue for drum transfer requests is maintained and as a drum transfer terminates the next queued request is initiated. While a job is halted during a magnetic or drum transfer, the central executive switches control to the next free job on the execution list, which is able to proceed. In the University of Manchester Atlas system, and therefore the simulation model, this transfer of control is inhibited during a drum transfer, if the halted job is of a higher priority than the next free job. In reference to the simulation model, this change of control is prevented whenever a tape job is halted during a drum transfer, since only one tape job may be on the execution list at one time.

The magnetic tape supervisor is entered whenever a job requests a tape transfer which may consist of several blocks. In the Atlas system,

a queue for tape transfer requests is maintained. As the tape request reaches the top of the queue, the tape transfer is initiated. Tape transfers may be overlapped with normal execution, providing the pages accessed are not those being transferred. However, if a block is already involved in a transfer the job requesting the transfer is halted and control passed on to the next job able to proceed. The simulation model is simplified by the approximation that for the job making the tape transfer a page involved in the transfer is invariably accessed. As a result, each tape job is halted during a tape transfer and control is passed on to the next free job if possible.

Lastly, an output routine is entered to transfer output for a job to the system output tape, which is used to buffer the output of all jobs. Output may occur during execution and automatically occurs at the completion of each job. However, to reiterate, the simulation model assumes no output during execution, instead all output blocks are transferred automatically upon the completion of the job execution in one long transfer. If a request for output occurs while the output tape is busy, the request is queued in the queue maintained for tape requests.

2.3.2 OPERATION OF SIMULATION MODEL

This section describes the flow of jobs through the simulation model during processing. Each job processed to completion passes through

several well defined phases. In order, these are: generation, assembly, execution and removal. Figure 2.3 illustrates the flow of jobs through the simulation model. During processing of a job, that job may appear as an entry on only one of the four lists:

(1) the generation list;

(2) the job list or queue for assembly;

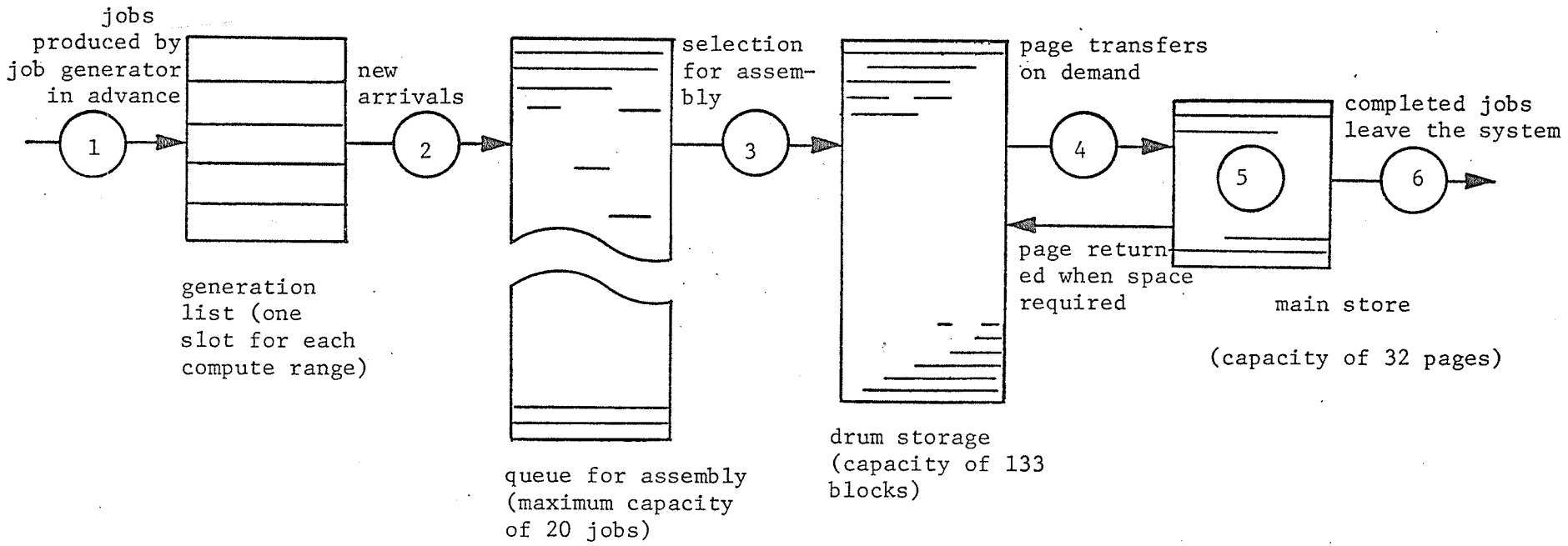
(3) the assembly list; and

(4) the execution list.

The generation list which is comparable to the Atlas input stream, consists of L entries made by the job generator. Each entry represents a job which is due to arrive. When the simulated time is equal to the arrival time, A_{ij}, of a job in the generation list that job is transferred to the assembly queue or job list. This transfer represents the entry of a job on the job list by the input supervisor in the Atlas system.

The job list or assembly queue represents a list of n_q entries ($n_q = 20$) comprising all jobs which are candidates for assembly. Each job in this queue competes for assembly and is selected by the job scheduler only when required by the simulated system. Jobs are queued into one of three conceptual queues or streams for tape, short and long jobs. As a vacancy appears on the assembly list, the appropriate queue is scanned, and by means of the job scheduling algorithm, the next job is chosen for assembly from the candidates available. Once chosen, a job is removed from the assembly queue and is entered on the assembly

FIGURE 2.3 JOB FLOW THROUGH SIMULATION MODEL



1. The job generator is discussed on pages 9 to 23.
2. New arrivals are selected as soon as they are due providing there is space in queue.
3. The algorithm used to select jobs for assembly is given on pages 43 and 44.
4. The discussion of page transfers is given on pages 49 to 51.
5. Only two jobs are in the execution phase at any given time and the distribution of the CPU between these two jobs is discussed on pages 55 to 58 . (See Figure 2.8).
6. As soon as the required amount of CPU time has been accumulated and the output transfer completed, removal operations are initiated (page 48).

list. The empty slot of the assembly queue is again available for the next job that has arrived. The simulation model assumes that no overhead is incurred for the transfer of a job from the assembly queue to the assembly list. The actual overhead being in the order of a few microseconds is considered negligible in this study.

The assembly list of the simulation model consists of a single entry slot. When the assembly slot is vacant, the required job is entered on the assembly list by the job scheduler. The algorithm which selects a job for assembly includes logic to verify that the system resources (magnetic tapes and storage) may be allocated to the job considered for assembly. Once entered on the assembly list, the job remains on this list for a minimum duration equal to its assembly time, t_{ai}, or until that time the job may be accepted onto the execution list. The assumed assembly time is taken to be a function of the job storage size and represents the time taken to transfer a job from the system input tape through core storage to drum storage. For each job i, the assembly time is given by,

$$t_{ai} = R_{ij}(d_r + t_r) \quad , \qquad \text{.....(2.3.1)}$$

where R_{ij} is the storage blocks requested by job i, d_r and t_r denote the drum and tape response times, respectively. The model is somewhat simplified in that the actual transfers required to store a job onto drum storage are ignored and the drum is made available for drum transfers for executing jobs.

As jobs on the execution list are completed the execution slot is freed and the assembly list is scanned for an assembled job. If found, the assembled job is transferred to the execution list from the assembly list and the assembly slot made available for the next job that the system requires to be assembled. The simulation model incurs a supervisor overhead of 0.006 seconds for the transfer of the assembled job from the assembly list to the execution list and the organizing of the job for execution by the supervisor (Central Executive).

Lastly, the execution list consists of n_e slots ($n_e = 2$) and contains those jobs which are being executed, that is the central processor is allocated to jobs appearing on the execution list. In the simulation model, upon completion of the job entry onto the execution list the storage blocks for that job are allocated with the first block located in a page of core storage and the remaining $(R_{ij} - 1)$ blocks on drum storage. A block directory corresponding to each execution list resolves the physical location of each block in storage. Jobs are placed into the first vacant slot available on the execution list as each execution slot is 'identical'.The simulation model does not involve roll-in, roll-out operations, although during the execution almost an entire job may be transferred back onto drum storage. For each job on the execution list, at least one page is retained in core storage for its entire duration. Thus, once an execution slot is allocated to a job, it remains allocated to that job until the job has received its full requested compute time and has completed its output transfer. The job is removed from

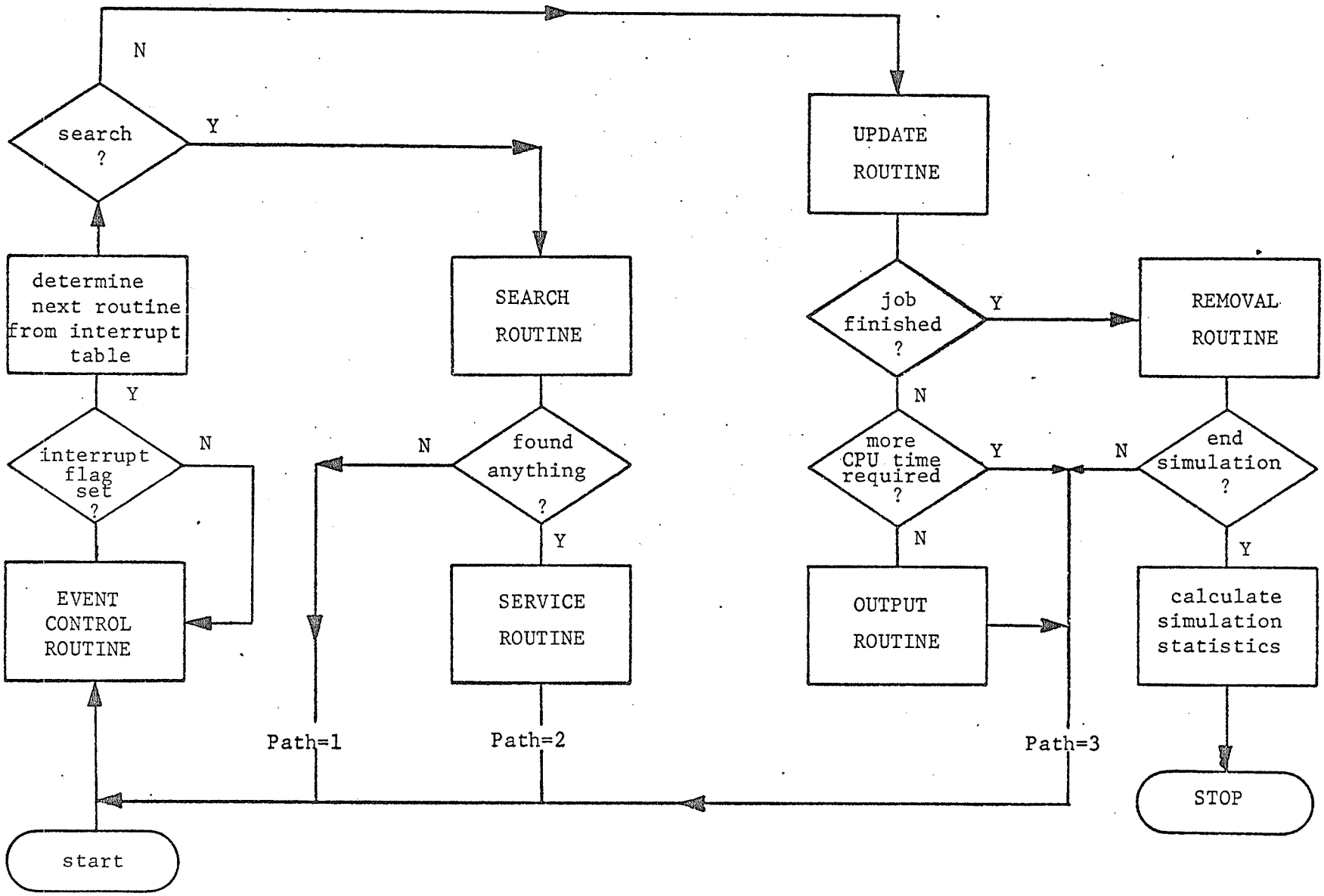
the execution list upon the completion of the output transfer and its execution slot made available to the assembled job. Under certain conditions, the execution list may be empty or partially filled. These conditions reflect a sparse load on the simulation system or a heavy demand on system resources, preventing assembly of the next available job.

During a simulation run, the series of transfers from generation list to the assembly queue, from assembly queue to the assembly list and from assembly list to execution list continue in this order, until a total of n jobs have been processed to completion. When n jobs have been processed through the system, the simulation statistics are calculated and the simulation run terminated. The generation list, job list or assembly queue, assembly list and execution list are not flushed. With each of the four defined lists, an associated sub-list is maintained containing the variables which describe each job and the relevant data generated and accumulated during the processing of each job. A description of the data maintained in each list for each job is given by Appendix A.

2.3.3 DESCRIPTION OF THE MODEL

The simulation model developed consists of seven major or primary routines: event control, switching, search, service, update, output and removal routines, each described in order. Figure 2.4 presents a flow chart of the logic of the simulation model.

FIGURE 2.4 FLOW CHART OF SIMULATION MODEL



(1) The event control routine (Figure 2.5) may be considered as the 'core' of the simulation model. Its fundamental function is to co-ordinate and organize in a logical sequence, the events occurring during the processing of jobs by the model. It is convenient for the purpose of description to present the event control in the various logical parts of which it consists. The event control routine, in addition to containing the previously described job-stream generator, performs the selection and entering of jobs onto the assembly, job and execution lists. When a job is required to be assembled, the job scheduler algorithm, part of the event control routine, selects the next job to be assembled and initiates assembly of that job. Assembly of jobs occurs concurrently with normal processing and once initiated, the event control routine advances the time to the next event to occur at a time which is the lesser of:

(a) the time of arrival of the earliest job onto the job list or assembly queue;

(b) the time when assembly of a job in the assembly phase is completed;

(c) the time of the next terminating interrupt due to the central processor or drum and magnetic tape transfers;

(d) the time of arrival of the next request for a drum or magnetic tape transfer for an executing job; and

FIGURE 2.5 THE EVENT CONTROL ROUTINE

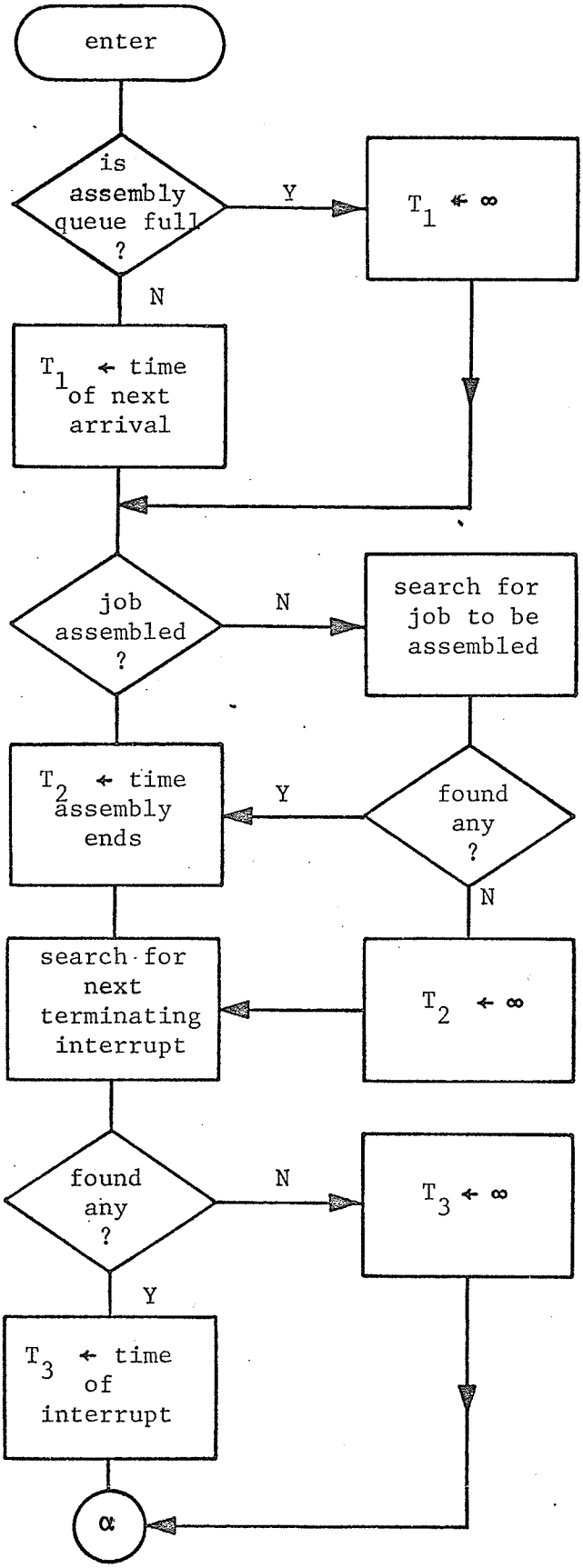


FIGURE 2.5 EVENT CONTROL ROUTINE (continued)

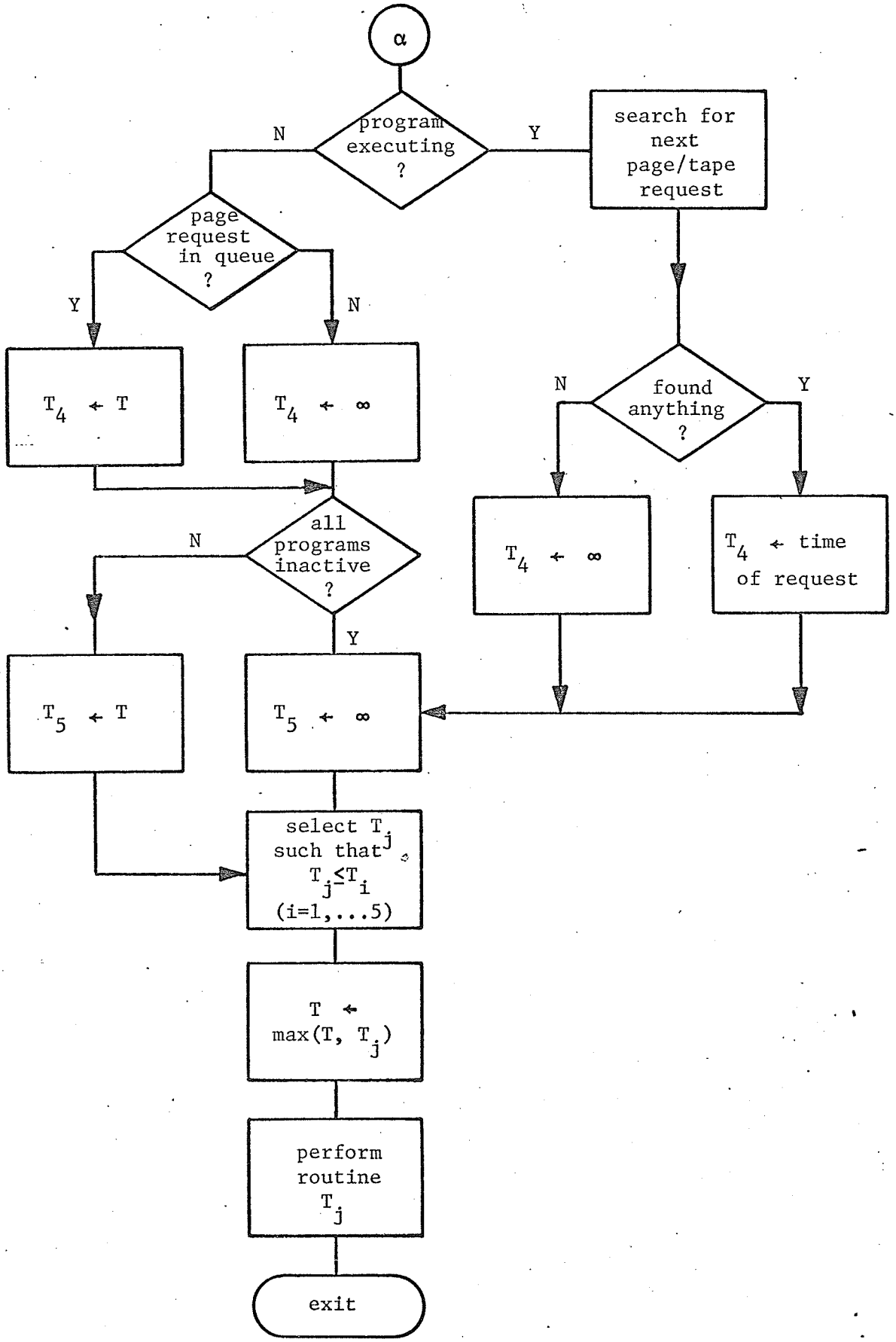


FIGURE 2.5 EVENT CONTROL ROUTINE (continued)

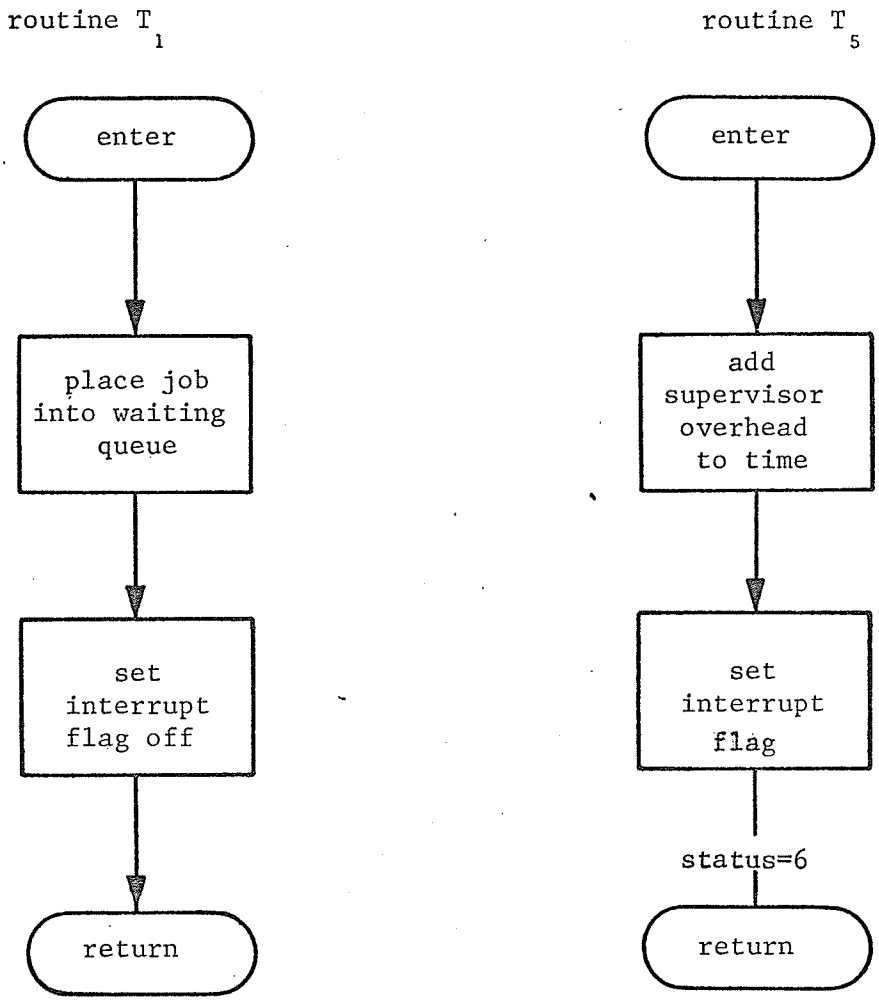


FIGURE 2.5 EVENT CONTROL ROUTINE (continued)

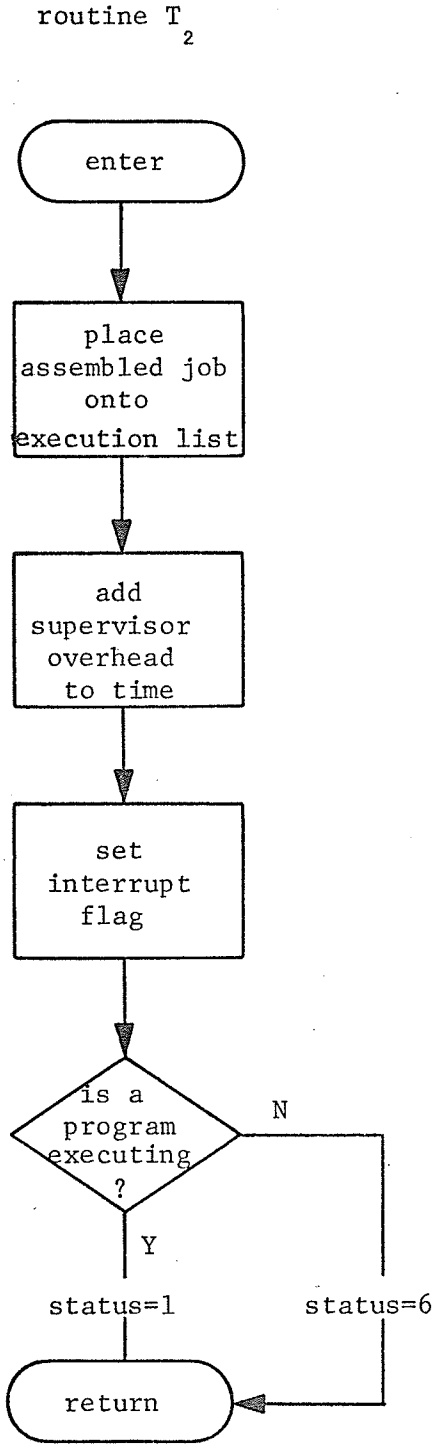


FIGURE 2.5 EVENT CONTROL ROUTINE (continued)

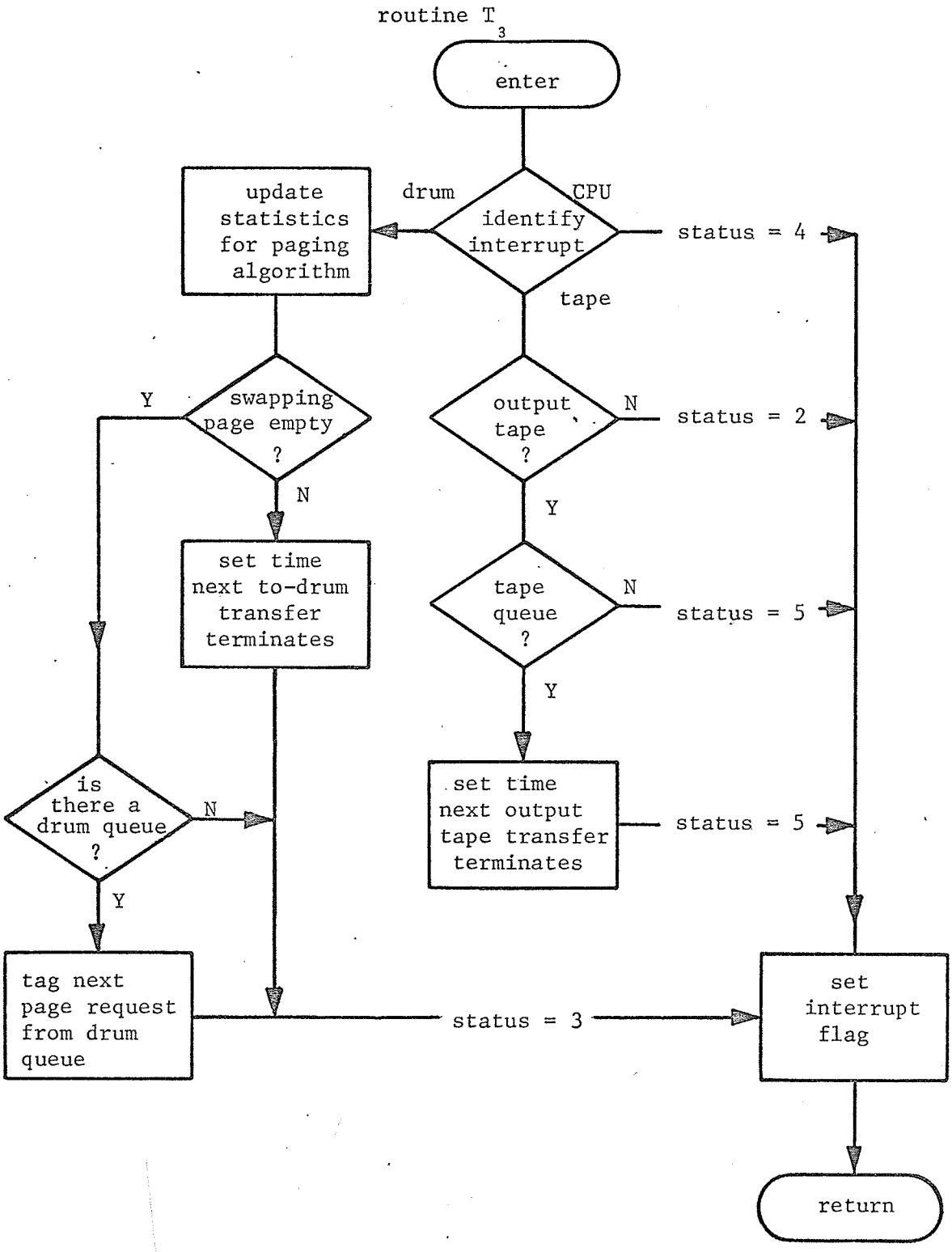
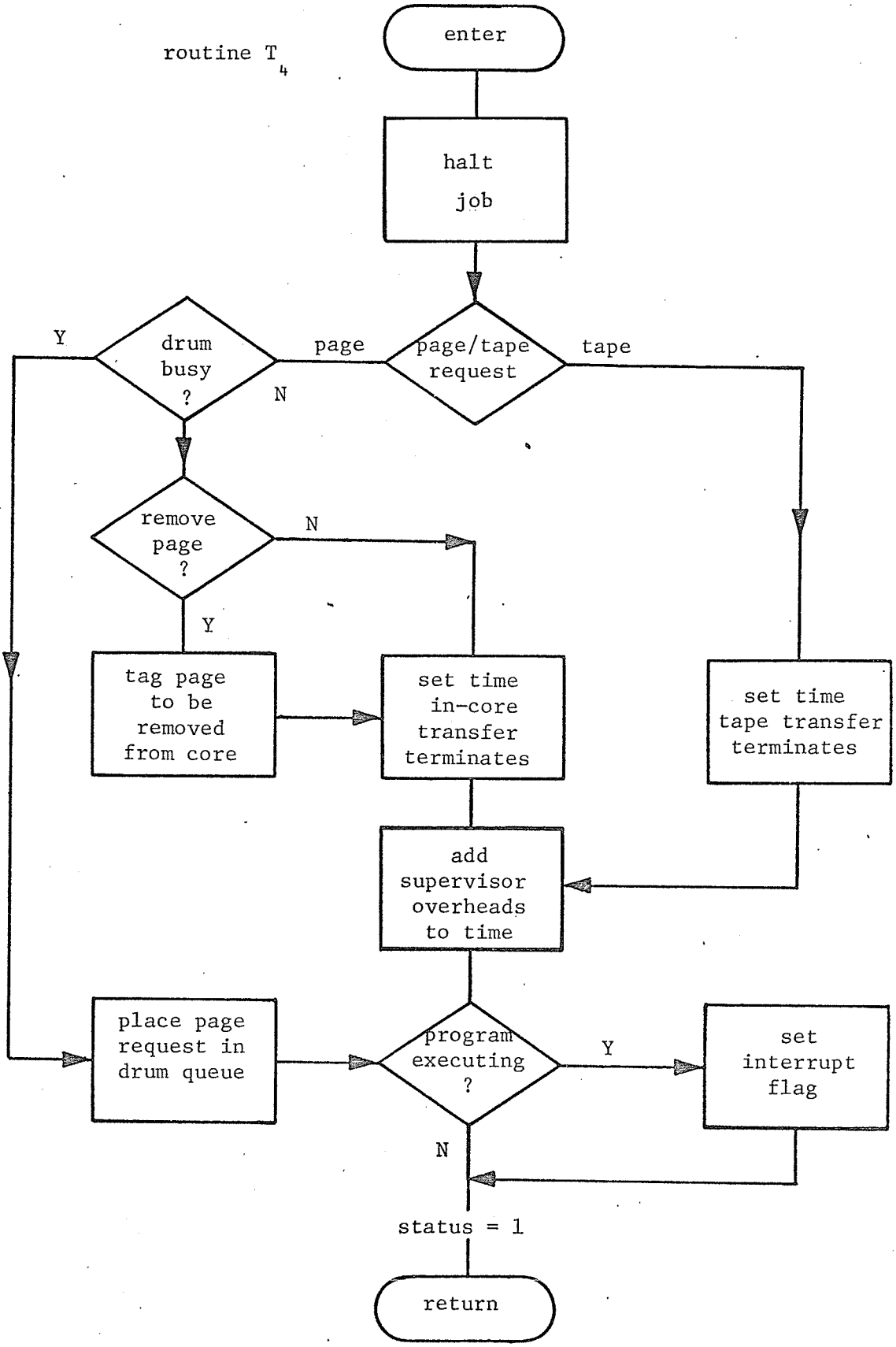


FIGURE 2.5 EVENT CONTROL ROUTINE (continued)



(e) the current time if no job is executing and if a useful operation could be performed after a subsequent supervisor cycle, for example, the initiation of execution for a job.

Events are processed on a first-come-first-served basis. However, in the event of a coincidence an arbitrary priority rule is invoked in the model. This priority rule is taken to be the precedence order of events type (a), (b), (c), (d) and (e). It should be stressed that the occurrence of a coincidence in event times is somewhat rare since the times are stored as normalized binary numbers with a 24 bit fractional part.

Corresponding to each event, the event control routine contains a piece of logic or sub-routine which performs the required action once the next event is chosen. These may be summarized as:

(a) the logic which transfers a job from the generating list into the assembly queue where it awaits selection for assembly.

(b) the logic which enters a job from the assembly list into the first available execution slot. As a job is entered onto the execution list, storage for that job is resolved in the block directory associated with each execution slot and magnetic tapes, if any, assigned. The availability of these resources were established before the job was selected for assembly. Storage for each job is allocated

such that the first page of the job is placed into core storage and the remaining blocks distributed in vacant blocks of drum storage. The actual transfers to locate these blocks on drum storage are not performed by the model. User magnetic tapes are dedicated to a job occupying an execution slot and remain assigned to that job for its entire duration. The supervisor overhead time due to this operation is taken to be 6 milliseconds which represents the average time the Atlas supervisor spends in the Central Executive [12]. A completed entry on the execution list signifies that a job is now ready to receive attention from the central processor.

(c) the logic which handles the terminating interrupts. This logic may be entered for three types of interrupts. These interrupts arise from the completion of a drum and magnetic tape transfers and the completion of a job's compute time. Coincident interrupts are given priority in the precedence order of drum, tape and CPU. The actual supervisor overheads due to hardware functions and switching between supervisor routines are to the order of a few microseconds. In the simulation these overheads are ignored.

(d) a piece of logic which accommodates the drum and tape transfer requests for an executing job. This logic performs the necessary housekeeping for paging operations and enters drum requests into the drum queue if necessary. If it is necessary to transfer a page from core upon completion of a page transfer to core storage, the simulated drum-learning algorithm contained in this logic selects and marks the page to be transferred out to drum storage. It is necessary to transfer a page out of core whenever core storage does not contain a free page for page swapping.

(e) the logic which is executed when the central processor is idle and a useful operation could be performed on the next supervisor cycle. If on the next cycle, a job was found to be free to receive attention from the central processor, this supervisor cycle represents the time required to switch control between jobs and is taken to be 2 milliseconds [12].

The job scheduling algorithm selects from the assembly queue the next job to be assembled. As the simulated system requests the assembly of a certain type of job, (long, tape, short) the job scheduler searches on a first-come-first-served basis for the next job which may be assembled amongst the possible candidates. As a completely assembled job has its

storage blocks located on drum storage and magnetic tapes mounted, the job scheduler algorithm scans for a job for which these requested system resources are available prior to selection for assembly. The rules which govern the type of job required by the system as presented [18] may be summarized by:

(a) search for long job, if execution list is empty;

(b) search for tape job, if execution list is void of a tape job;

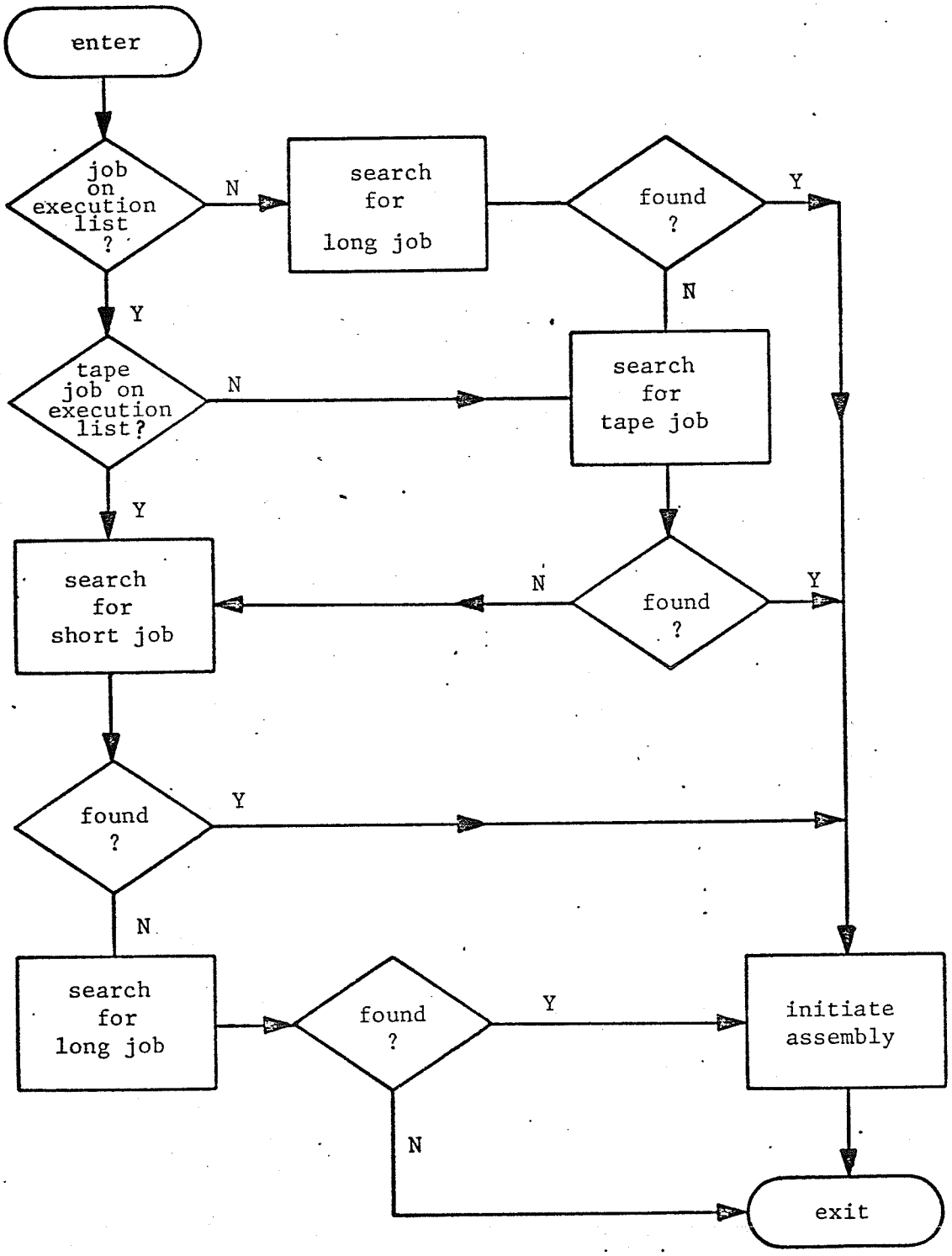
(c) search for a short job; and

(d) search for a long job if execution list is void of a long job.

The algorithm ensures that tape and long jobs are processed serially. However, short jobs are processed independently from both long and tape jobs. Once selected, assembly of a job is initiated and occupies the assembly slot for a minimum duration equal to its assembly time. Figure 2.6 illustrates the flow logic of the job scheduler used in the simulation model.

(2) The switching routine determines which routine is to be executed next, the search or update routine. The selection is dependent on which one of the three paths was last executed and the type or status of the terminating interrupt. Interrupts represent the completion of a supervisory task, upon which subsequent updating or searching is required. The switch table used to determine each operation to be performed

FIGURE 2.6 JOB SCHEDULING ALGORITHM



upon each interrupt is shown in Table 2.2. Vacant entries appear, because certain combinations of interrupts and last paths executed do not occur. For example, if the previous cycle involved the search routine (path = 1, Figure 2.4), indicating a job is not executing, then upon leaving the event control routine, the status of an interrupt could not be 1 or 4, indicating a job was interrupted during execution. Therefore, it is not possible for an 'update CPU interrupt' to occur with path 1, and the same argument holds for path 3.

(3) The search routine locates a job on the execution list which has not yet received its full requested CPU time and is ready to receive more attention from the central processor. A straight search is performed on the first-found-first-served basis for a free job which ranks greater or equal in priority to any of the other jobs on the execution list. For example, if two jobs on the execution list are both ready and are of equal priority, the job entered in the first slot of the execution list would be selected by the search routine.

(4) The service routine initiates execution for the job located by the search routine.

(5) The update routine records the various combinations of accumulated CPU time, CPU and drum device time, CPU and magnetic tape device time. When a job has received its full requested CPU time, the output routine is activated. If a job is finished, that job is tagged for removal from the simulated system.

TABLE 2.2

SWITCH TABLE CONTAINING THE DESCRIPTION OF THE NEXT ROUTINE

Status \ Path	1	2	3
1	---	update CPU time	---
2	update tape device time	update tape device and CPU time	update tape device time
3	update drum device time	update drum device and CPU time	update drum device time
4	---	update CPU time (execution finished)	---
5	remove job	update CPU time and remove job	remove job
6	search routine	---	search routine

The entries in the above switch table indicate the functions or routines to be performed upon an interrupt.

(6) The output routine, activated by the update routine, initiates the output block transfer onto the system output tape. If the system output tape is busy, the request is queued.

(7) The removal routine releases all system resources allocated to the job tagged by the update routine and removes that job from the simulated system. The time of completion is recorded and all relevant data pertaining to the completed job is collected.

In summary, the system operational parameters used to control the operations of the simulation model include:

n_g, the number of generation slots;

n_e, the number of execution slots;

n_q, the number of slots in assembly queue;

n_d, the number of slots in drum queue;

n_t, the number of slots in tape queue (output tape);

d_r, the drum response time;

t_r, the magnetic tape response time;

C, the number of pages of core storage;

D, the number of blocks of drum storage;

mt, the number of user magnetic tapes;

t_{s1}, the supervisor overhead for central executive; and

t_{s2}, the supervisor overhead to switch between jobs.

2.4 PAGING IN ATLAS SYSTEM AND THE MODEL

This section describes the paging operations in the Atlas computer and how these operations are simulated in the model. In the following description, emphasis is placed on the drum-learning program. The drum-learning program reflects the logic of the replacement strategy used to select the page to be transferred to drum storage in order to maintain an empty swapping page. A brief discussion of the Atlas drum-learning program is included only to make the drum-learning algorithm used by the simulation model apparent. Details concerning the drum-learning program and the storage organization in the Atlas computer have been presented elsewhere [17].

2.4.1 ATLAS DRUM-LEARNING PROGRAM

In the Atlas computer, paging is used as a means of storage management for each job. Jobs are segmented into manageable sections or pages each dynamically relocatable during execution. Administrating hardware, records data on the page usage, which interrupts the supervisor if a required page is not found in core storage. In this manner, pages are only loaded into core storage as they are required by the executing job. When a page is required from drum and core storage is full, a page is swapped back onto drum storage in order to maintain an empty page for the next page demand. The replacement strategy, to select the page to be transferred, is based on the 'drum-learning program'. This drum-learning program records information on the length of time that each

page currently in core storage is accessed and the previous duration of inactivity for that page. Using this information, by a simple algorithm, the learning program attempts to find a page which appears no longer in use. However, if all pages are currently in use, it predicts the page that will be the last required, if the current page usage pattern is maintained. In this manner, the drum-learning program attempts to detect a loop pattern of page references and tries to maximize the time between page transfers. Approximately 99.99% of all the pages requested are found to be already in core storage [10].

2.4.2 SIMULATED DRUM-LEARNING ALGORITHM

As a job is entered on the execution list only the first block of that job is located in core storage, with the remaining blocks located on drum storage. In this simulation, the page references to pages in core are ignored and only the out-of-core page requests are generated. The assumption is made that in-core page references for each job i are normally distributed about $R_{ij}/2$, where R_{ij} is the number of storage blocks requested. It follows then that for any page reference the further away from this assumed mean, the greater the time before this particular page will be required. During execution of a job i, the out-of-core page requests occur at a mean rate of $\overline{\delta}_j$ for each job belonging to compute range j. The particular page to be transferred into core is

randomly chosen from the remaining pages on drum storage. However, the page transferred back to drum storage is that page furthest away from the assumed mean $R_{ij}/2$ for each job i on the execution list. In this manner a loop pattern of page usage is established clustered about an arbitrary mean of a normal distribution. The in-core pages located furthest away from this mean represents those pages which will be last required by the executing job.

2.5 PERFORMANCE CHARACTERISTICS

There are two independent statistics used to characterize system performance, the mean relative response $\overline{\omega}_j$ and the mean duration $\overline{\Delta T}_j$. The characteristic $\overline{\omega}_j$ adopted in this research is similar to the relative response ω used by Fife [19] and Nielson [2]. The assumption is that if values of $\overline{\omega}_j$ and $\overline{\Delta T}_j$ (j = 1, 2, 3 and 4) are reproduced by the simulation model, the simulation is accurate.

The relative response, ω_{ij}, of a job in the job-stream of compute range j is defined to be the ratio of the total time elapsed between completion of assembly to job completion and the sum of the magnetic tape and drum device times and the CPU time received. The time interval from job assembly to completion, excludes the assembly queue wait time and represents in accordance with the Atlas computer the time for which each job is monitored.

The duration of any job i is taken to be the time elapsed between the end-of-assembly time, A_{ij}^*, and the job completion time, T_i, and is given by:

$$\Delta T_i = T_i - A_{ij}^* \quad , \qquad .....(2.5.1)$$

where the end-of-assembly A_{ij}^* may be defined as:

$$A_{ij}^* = A_{ij}' + t_{ai} \quad , \qquad .....(2.5.2)$$

where A_{ij}' is the time when assembly of job i is initiated and t_{ai} denotes the assumed duration of assembly for that job.

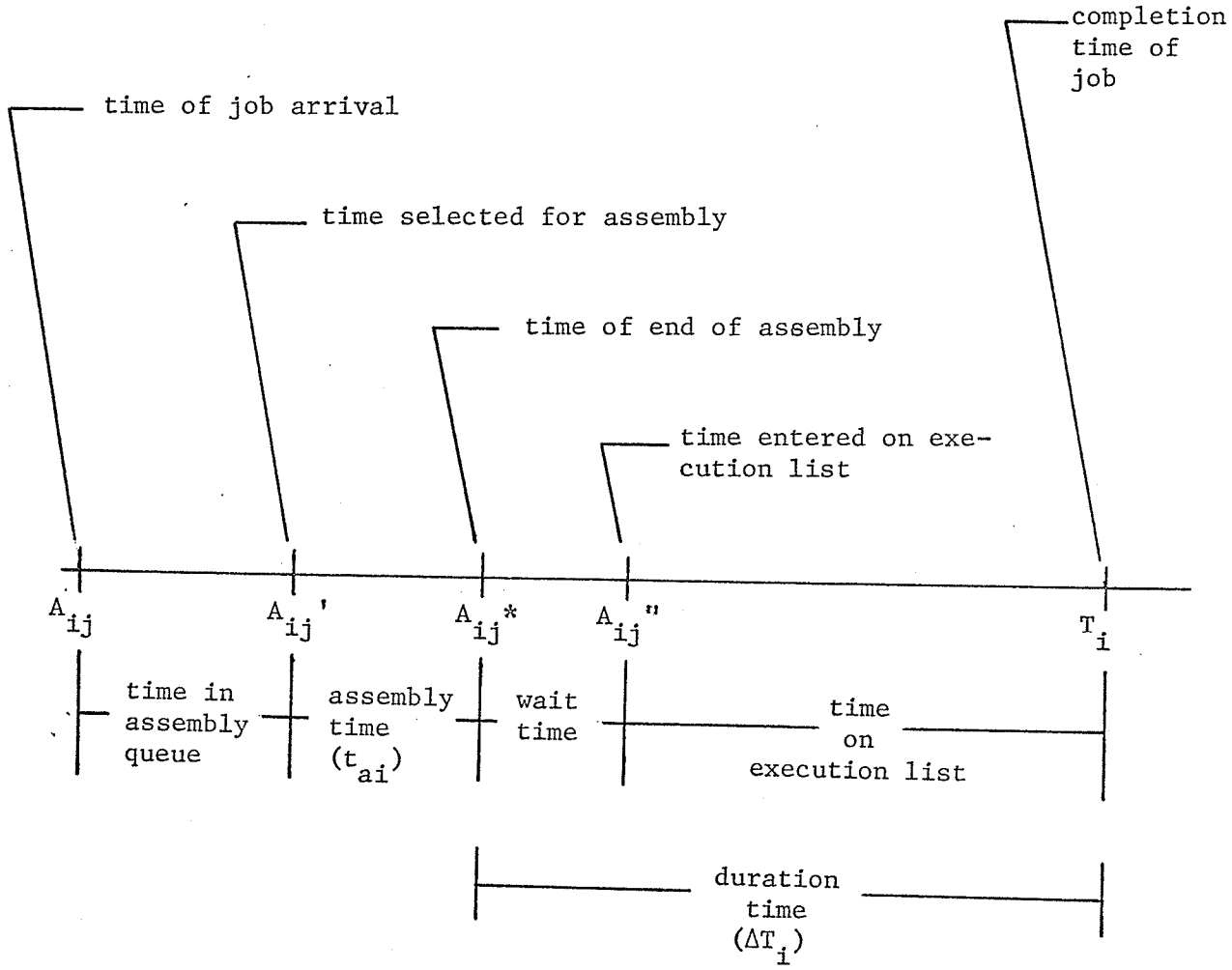
The earliest time that a job may be entered onto the execution list is A_{ij}'' if an execution slot is immediately available, then $A_{ij}'' - A_{ij}^*$. The time interval defined by $A_{ij}'' - A_{ij}^*$ is the time an assembled job must wait before it may be entered on the execution list. The average wait time for jobs of each compute range is output by the simulation model. Figure 2.7 illustrates the time relations described above. Each job i in the job-stream is assumed to receive some drum device time, d_i, due to paging and magnetic tape device time, t_{mi}, due to system or user magnetic tape transfer. The earliest time that a job may be completed, ignoring supervisor overheads approaches the value:

$$T_i \simeq A_{ij}^* + \xi_{ij} + d_i + t_{mi} \quad . \qquad .....(2.5.3)$$

The actual value of T_i, in fact, may be slightly less, since some of the drum device time may be overlapped with execution of that job. The

FIGURE 2.7

PROCESSING TIMES DIAGRAM



For minimum duration, ΔT_i min;

wait time = 0, ($A_{ij}'' = A_{ij}*$)

minimum time a job will be on the execution list is seen to approach the value given by:

$$\Delta T_i \simeq \xi_{ij} + d_i + t_{mi} \quad . \qquad .....(2.5.4)$$

Time on, for the monitoring of each job, begins at the time of end-of-assembly, A_{ij}*. The duration of each job is a measure of time for which system resources (storage, tape, drum and CPU) are occupied by that job and includes the wait time required before a job may be entered on the execution list. Thus, the mean duration, $\overline{\Delta T}_j$, for the n_j jobs of compute range j which are given in the Atlas statistics and calculated by the simulation model may be used as an additional characteristic of performance.

At job completion, the sum of the magnetic tape and drum device times plus the central processor time devoted to a job i is denoted by:

$$\tau_i = \xi_{ij} + d_i + t_{mi} \quad . \qquad .....(2.5.5)$$

The relative response, ω_{ij}, of job i is a measure of how rapidly the system responds in the processing of that job and is defined as:

$$\omega_{ij} = \Delta T_i / \tau_i \quad , \qquad .....(2.5.6)$$

similarly the minimum relative response of a job is defined as:

$$\omega_{ij} = \frac{\Delta T_{i\ min}}{\tau_{i\ min}} \simeq \frac{\xi_{ij} + d_i + t_{mi}}{\xi_{ij} + d_i + t_{mi}} \simeq 1. \qquad .....(2.5.7)$$

The mean relative response, $\overline{\omega}_j$, for each compute range j is determined by:

$$\overline{\omega} = \frac{1}{n} \sum_{i=1}^{n} \omega_{ij} \quad , \qquad \text{.....(2.5.8)}$$

where n is the total number of jobs processed in the simulation run.

The CPU utilization, E, is defined to be the ratio of the simulated time T that the CPU is active on users jobs. This time includes only that compute time ξ_{ij} for each job i and excludes all waits for magnetic tape and drum transfers and supervisor overheads.

The CPU utility is determined as:

$$E = \sum_{i=1}^{n} \xi_{ij}/T \quad , \qquad \text{.....(2.5.9)}$$

where ξ_{ij} is the compute time job i and T the duration for the processing of all the n jobs.

2.6 ALLOCATION OF CENTRAL PROCESSOR

The central processor is dynamically time-shared between a maximum of two jobs on the execution list and the supervisor. Thus, once the central processor has been allocated to a given job on the execution list, that job continues execution until that time when it is suspended to service an interrupt. Upon completion of the service of such an interrupt, the search routine of the simulation model again searches for a job to which it may allocate the central processor. Interrupts occurring from time to time, that are of interest in this discussion, are those resulting from the arrival of a new job onto the execution list

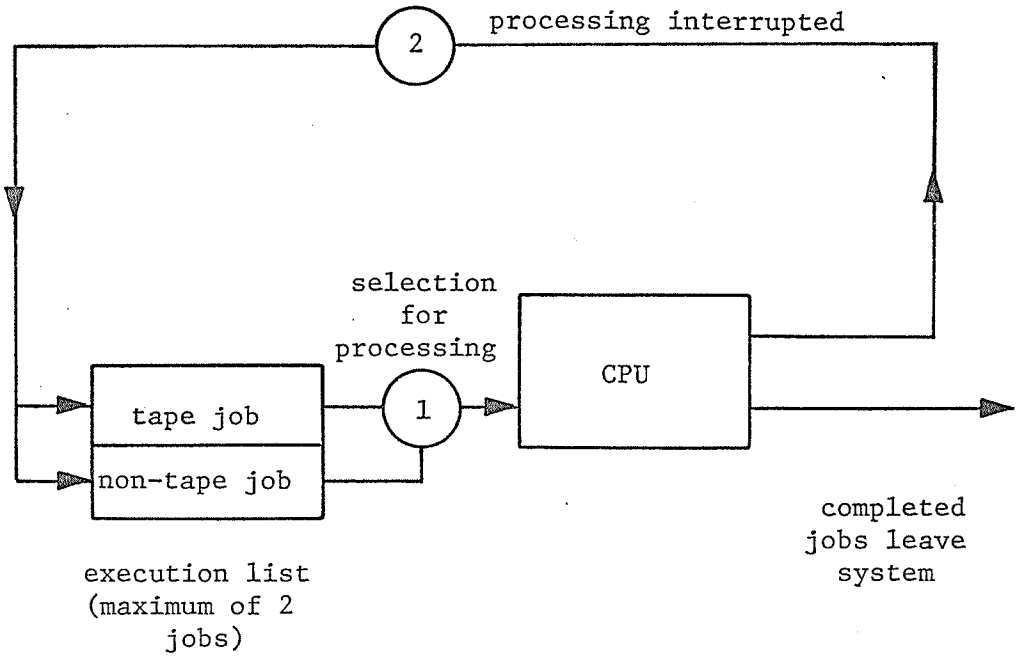
and those resulting from drum and magnetic tape transfers. Figure 2.8 illustrates the above discussion.

If the executing list is shared between a magnetic tape job and a non-tape job, the magnetic tape job having the higher priority seizes the central processor until a request for a tape or drum transfer occurs for that job. In this situation, the central processor may be allocated to the non-tape job, only during magnetic tape transfers of the tape job. Thus, the non-tape job must wait for each tape transfer to receive its compute time, as long as the tape job remains on the execution list.

The situation is somewhat different if the execution list is shared between two non-tape jobs since each has equal priority. Under these conditions, the central processor may be allocated to each job in an alternating pattern as each job is halted during a drum transfer. Because drum queues may arise, both jobs on the execution may be in a halted state, with the central processor remaining idle.

It is interesting and apparent that the allocation of the central processor and the duration for which it is allocated depends on three factors. First, allocation of the central processor depends on the priority of jobs on the execution list, tape jobs having the highest priority. Secondly, allocation of the central processor depends on the physical location of a job on the execution list, as the search routine accepts jobs of equal priority, in the order of first-found-first-served. This is a simplification and an artifact of the simu-

FIGURE 2.8 FLOW LOGIC OF CPU ALLOCATION



1. The algorithm (SEARCH ROUTINE) determining selection for processing is given on page 46 .

2. The conditions under which processing is interrupted are given on page 42.

lation model. Lastly, the length of time the central processor is allocated to any one job depends on the distribution of the inter-arrival time of both the drum and magnetic tape transfers. Because of these requests, an interrupted job is unable to proceed and allows the central processor to be allocated to another job if one is ready to proceed. Appendix D gives some example profiles of interaction between the central processor, supervisor, magnetic tape and drum transfers.

2.7 DISCUSSION OF THE SIMULATION MODEL

The assumptions and approximations used in this simulation have been discussed earlier. It is convenient at this time to summarize and contrast some of the major features of the simulation model to the Atlas system.

The University of Manchester operates the Atlas computer in alternate periods of time of 2 to 3 hours throughout each day. This may necessitate that the system is flushed upon completion and restarted upon initiation of each period of operation. In contrast, the simulation model assumes that operation is continuous over a week taken to be 43 hours.

It is apparent that intervention in the operation of a computer by the computer operator may affect the behaviour and performance of the Atlas system. However, these interventions are not subject to simula-

tion without an exact knowledge of their nature. For this reason, operator assigned priorities and streaming of jobs before entry into the system are not considered.

In the Atlas system during execution of a job, output for a job may occur at any particular moment during execution and automatically upon completion. Again, the model simplifies the real conditions and assumes that all output observed for a job occurs upon completion of execution.

During execution of a job the Atlas system incurs numerous overheads resulting from both system hardware and supervisory tasks; for example, overheads due to switching between supervisor routines, initiating peripheral devices and transferring job between various lists. Overheads of magnitude less than 1 millisecond are ignored by the simulation model.

Information concerning supervisor requests for magnetic tape and drum transfers was not collected for this study. Thus, these are ignored by the simulation model. As a result, the length of the drum queue is reduced from 64 entry slots [12] to 3. The maximum number of drum requests that may possibly be queued for a maximum of two jobs on the execution list of the simulation model is three.

A discussion of the Atlas performance [10] indicated that magnetic tape jobs accounted for approximately 16 per cent of the Atlas jobstream. However, the percentage of magnetic tape jobs was not calculated for the Atlas statistics used in this study and 16% was determined to be too sparse. All jobs are generated using Monte Carlo

techniques and it was possible to predetermine the percentage of tape jobs by a suitable rounding formula:

$$U_{ij} = U_{ij}^{*} + \zeta \quad , \qquad \text{.....(2.7.1)}$$

where $0 \leq U_{ij} \leq mt$ and $0 \leq \zeta \leq 1$. In the simulation model $\zeta = 0.5$ resulting in a simulated job-stream in which approximately 38% of the jobs are tape jobs.

To conclude, during execution of a job, magnetic tape rewinds, block transfers and out-of-core page requests occur at a rate which is time dependent. For example, the rate of out-of-core page requests tends to be much larger when a job has just started processing than when it has been processing for some time [10]. The simulation model however considers that these activities are time independent with the mean values input or derived from the input job-mix parameters. Also tape jobs within compute range j requesting m magnetic tapes have the same mean rate of block transfer requests.

In Chapter IV the problem concerning the rate of out-of-core page requests is approached in a more realistic manner. The out-of-core page request rate for each job is taken to be a function of the ratio of pages resident in core to the job's total storage size.

CHAPTER III

SIMULATION OF THE ATLAS SYSTEM

3.1 INTRODUCTION

In this chapter the reproducibility and the reliablility of the Atlas simulation results are discussed. The criteria for the characteristics of performance used in this research are presented in Chapter II. This chapter, then, investigates the deviation of the simulation results from the results observed in the Atlas output statistics. Repeated simulation runs, subject to similar conditions are performed using a typical set of Atlas job-mix parameters, with one parameter varied on each simulation run.

The Atlas computer processed an average of 2500-3000 jobs per week. Six sets of statistics collected over consecutive weeks are shown in Table 3.1.

3.2 CHOICE OF JOB-MIX PARAMETERS

The job-mix parameters for the simulation model obtained from the Atlas statistics were fully described in Chapter II. As the actual run time of the simulation model is highly dependent on the paging activity, a favourable job-mix in this study consists of one in which the mean observed drum device time, $\overline{d}_j$, is relatively low

T A B L E 3 . 1

UNIVERSITY OF MANCHESTER - ATLAS STATISTICS

WEEK ENDING 21/9/68 TO 26/10/68

	COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	COMPILE STORE (BLKS)	EXECUTION STORE (BLKS)	NO. OF TAPES	PRINTER OUTPUT (LINES)	BLOCKS TRANS. (BLKS)	ONE-LEVEL STORE (SEC)	DURATION (SEC)	RELATIVE RESPONSE TIME
(i)	<1	0.3	360.0	61.9	34.7	0.8	22.7	423.6	1.0	80.8	266.7
	1 - 8	3.8	864.0	65.3	37.3	0.4	132.5	344.2	6.3	37.7	2.5
	8 - 120	33.1	1133.0	77.4	55.5	0.5	732.1	449.6	21.6	102.2	1.6
	120 - 960	284.0	194.0	75.5	56.0	0.5	1132.4	452.5	54.6	456.9	1.3
	>960	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
(ii)	<1	0.3	366.0	57.8	33.5	0.8	27.0	369.0	1.3	57.2	65.2
	1 - 8	4.0	801.0	67.7	39.3	0.5	164.3	397.2	6.9	45.9	2.7
	8 - 120	32.5	1354.0	76.2	54.9	0.5	688.4	476.5	20.3	109.4	1.8
	120 - 960	324.3	277.0	79.3	61.7	0.6	1140.2	841.1	71.6	569.4	1.3
	>960	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE 3.1 (CONTINUED)

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO.OF JOBS	COMPILE STORE (BLKS)	EXECUTION STORE (BLKS)	NO.OF TAPES	PRINTER OUTPUT (LINES)	BLOCKS TRANS. (BLKS)	ONE-LEVEL STORE (SEC)	DURATION (SEC)	RELATIVE RESPONSE TIME
(iii) <1	0.3	349.0	57.3	34.8	0.6	28.1	255.2	1.3	37.7	133.7
1 - 8	3.9	909.0	67.3	39.3	0.5	149.2	256.3	6.0	39.5	3.2
8 - 120	33.1	1287.0	78.6	54.9	0.5	681.9	382.7	20.7	102.2	1.6
120 - 960	290.4	264.0	81.3	65.0	0.5	1072.5	807.4	92.4	551.2	1.3
>960	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
(iv) <1	0.3	418.0	55.8	32.6	0.6	29.2	313.2	1.4	58.1	180.2
1 - 8	3.9	833.0	69.3	40.2	0.5	173.7	284.1	4.9	38.5	2.4
8 - 120	32.6	1198.0	77.4	56.0	0.6	763.1	467.3	21.8	110.0	1.7
120 - 960	293.5	207.0	83.8	68.0	0.5	1088.4	633.1	97.9	541.5	1.4
>960	1229.1	2.0	70.0	57.5	1.0	274.0	567.3	6.5	1531.0	1.2
(v) <1	0.4	762.0	58.5	29.1	0.3	35.3	215.0	1.2	38.5	93.5
1 - 8	3.9	984.0	69.2	39.3	0.4	162.9	168.8	4.9	30.0	2.4
8 - 120	33.2	1318.0	78.5	54.7	0.6	726.0	447.9	24.6	113.1	1.7
120 - 960	266.5	250.0	85.9	65.8	0.7	1164.2	822.2	105.2	534.8	1.4
>960	1609.3	3.0	71.0	0.0	1.0	157.3	616.8	1.0	1591.0	1.0

TABLE 3.1 (CONTINUED)

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	COMPILE STORE (BLKS)	EXECUTION STORE (BLKS)	NO. OF TAPES	PRINTER OUTPUT (LINES)	BLOCKS TRANS. (BLKS)	ONE-LEVEL STORE (SEC)	DURATION (SEC)	RELATIVE RESPONSE TIME
(vi) <1	0.4	1013.0	56.7	29.2	0.3	40.8	111.3	1.2	26.1	68.9
1 - 8	3.5	1068.0	66.5	38.1	0.3	142.7	275.2	6.0	39.4	2.8
8 - 120	32.0	1385.0	77.3	54.0	0.6	742.5	482.6	22.7	111.4	1.7
120 - 960	306.9	281.0	82.2	65.4	0,7	1290.7	679.0	59.7	521.2	1.3
>960	1714.3	2.0	32.0	0.0	1.0	350.0	687.5	1.0	1740.0	1.0

in each compute range j. Aside from this consideration, the choice of a particular set of Atlas statistics is somewhat arbitrary. The particular job-stream to be simulated is represented by the Atlas statistics as shown in Figure 3.1 (i). These represent a typical Atlas job-mix with moderately heavy demands on the system.

Each job processed during a week of operation belongs to one of 5 compute ranges, (j = 1, 2, 3, 4 and 5). As shown, the bulk of the jobs in the job-stream are classified as short jobs and belong to compute ranges 2 and 3. The variation of the mean relative response for three compute ranges, (j = 2, 3, 4), in comparison to the remaining sets of Atlas statistics, is not significantly large. However, this is not true of the first compute range (j = 1) in which the range of mean relative response is as large as [65.2, 266.7]. This wide fluctuation in the mean relative response of compute range 1 suggests that this statistic is unstable and sensitive to one or more features of the Atlas system. These features may be totally or partially present in the simulations or excluded entirely. For example, these fluctuations may be a result of the particular job scheduling algorithm used or due to system restarts and operator assigned priorities.

3.3 REPRODUCIBILITY

Because the reproducibility problem is not solved [5], it is necessary to treat this problem in an *ad hoc* manner in order to reduce the majority of the uncertainty involved in the calculations.

Under the circumstances, it would be acceptable to simulate the operation of the Atlas computer over a period of one week and to repeat this calculation several times using a different pseudo-random number series. However, preliminary calculations indicated that each simulation would require 8 hours on a 360/65 computer (the machine available). This time is considered to be prohibitive for several such calculations.

Preliminary calculations based on pilot simulations also indicated that the variation of the run time with the rate of drum transfers was an increasing function with an increasing slope. Further, it was considered that calculations involving increasing rates of drum transfers (all other input parameters the same as in Tables 3.2 and 3.3) would extrapolate to the Atlas rate of drum transfers giving an extrapolated result equal in worth to several calculations using the full Atlas statistics. This would, of course, only be valid (i) if the direction of extrapolation was fairly well defined with a shallow slope, and (ii) if a different pseudo-random number series was used for each calculation contributing to the extrapolation.

The approach taken in this reproducibility study was to perform six simulation runs with the actual Atlas job-stream paging overheads reduced by a multiplying factor C_1 ($C_1 \leq 1$), for values .01, .1, .2, .4, .6 and .8. This was effected by assuming that the mean observed drum device time, $\overline{d}_j$, for each compute range j was not $\overline{d}_j$ but $C_1 \overline{d}_j$ where C_1 is a proportionality constant. In this manner, the rate of out-of-core page requests is decreased, reducing the overall paging overhead in the simulation model and therefore the machine time required to perform each run.

Trial simulation runs each involving in excess of 500 jobs for values of C_1 equal to .01, .1, .2, and .4 indicated that the plotted values for the mean relative response against C_1, assumed a reasonably smooth graph suitable for extrapolation. Simulation runs were continued for the values of C_1 equal to .01, .1, .2, .4, .6, and .8 until in excess of 800 jobs were processed in each case.

Figure 3.1 illustrates graphically the run times required to perform each run for 100 jobs at various values of C_1. The total time taken for the extrapolation scheme was approximately 6.9 hours. The run time increases linearly with C_1 and the intercept at $C_1 = 0$ indicates the speed at which the processing of jobs could be achieved without the simulation of page demands.

The simulation operational and input job-mix parameters used for each run were identical (except for C_1) and the same starting random number was used to initiate the random number generator. The fixed job-mix parameters and simulation operation parameters used are shown in Table 3.2 and Table 3.3, respectively.

F I G U R E 3 . 1

RUN TIME FOR 100 JOBS AS A FUNCTION OF C_1

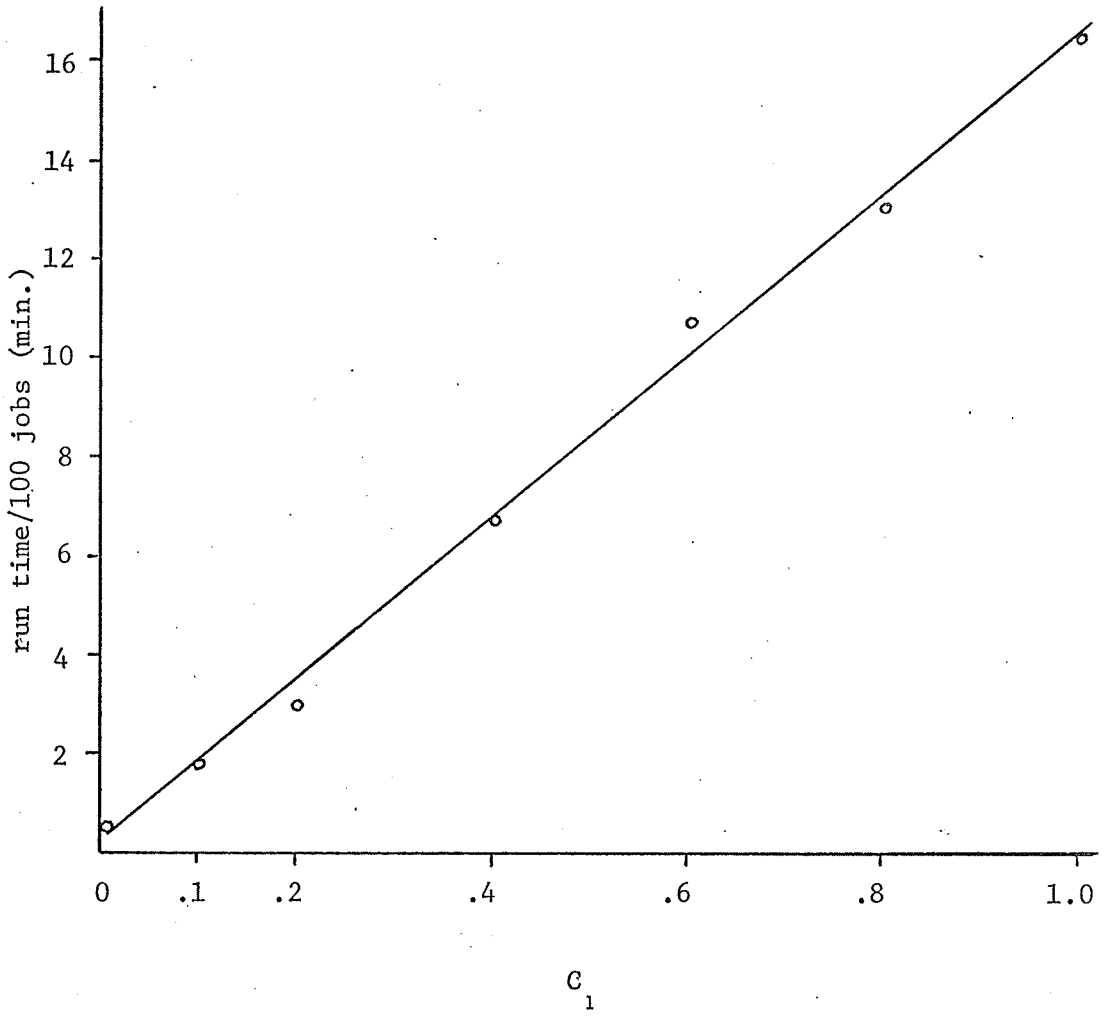


TABLE 3.2

INPUT JOB-MIX PARAMETERS USED FOR REPRODUCIBILITY STUDY

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANSFERRED (BLKS)	ONE-LEVEL STORE (SEC)	PRINTER OUTPUT (BLKS)
0 - 1	0.3	360.0	48.3	0.8	423.6	1.0	1.4
1 - 8	3.8	864.0	51.3	0.4	344.2	6.3	8.3
8 - 120	33.1	1333.0	66.5	0.5	449.6	21.6	45.2
120 - 960	284.0	194.0	65.8	0.5	452.5	54.6	70.8

TABLE 3.3

OPERATIONAL PARAMETERS USED FOR REPRODUCIBILITY STUDY

Number of generation slots,	$n_g = 4$
Number of execution slots,	$n_e = 2$
Number of assembly queue slots,	$n_q = 20$
Number of drum queue slots,	$n_d = 3$
Number of tape queue slots,	$n_t = 2$
Drum response time (sec),	$d_r = .014$
Tape response time (sec),	$t_r = .063$
Size of core storage,	$C = 32$
Size of drum storage,	$D = 133$
Number of user magnetic tapes,	$mt = 5$
(Central executive) supervisor overhead (sec),	$t_{s_1} = .006$
(Switching) supervisor overhead (sec),	$t_{s_2} = .002$
Proportionality constant,	$f = 2.00$
Paging load proportion,	C_1 varied
Starting random number,	= 568976679

It is important to point out that although the same initial value is used as a 'seed' for the random number generator, a different sequence of random numbers result for each run. In fact, the paging load alters the sequence of events occurring in each simulation run producing a different sequence of random numbers.

Clearly the Atlas job-stream is defined when C_1 is unity. Extrapolation techniques were used since the machine time required to process a sufficient number of jobs (with $C_1 = 1.0$) to obtain reasonably stable and reliable results is costly.

3.4 RELIABILITY OF SIMULATION RESULTS

The discrepancy between the simulation results and results logged by the Atlas computer is a measure of the reliability of the simulated results and hence the validity of the approximations and assumptions used in the design of the model.

Output of the simulation model consists of both input (generated by job generator) and output parameters for the processed job-stream. Since the input parameters such as the mean requested storage blocks, magnetic tapes, compute time and printer output are independent of time little discrepancy is expected in these parameters. However, the output parameters such as the mean duration and relative response are dependent on time, the scheduling of

jobs and allocation of the CPU. These parameters reflect the accuracy of the design of the simulation model and accordingly are used to examine the reliability of the simulation results. The input and output parameters of the Atlas statistics are shown in Table 3.1 and the simulation results are the results of the extrapolation scheme discussed in the previous section.

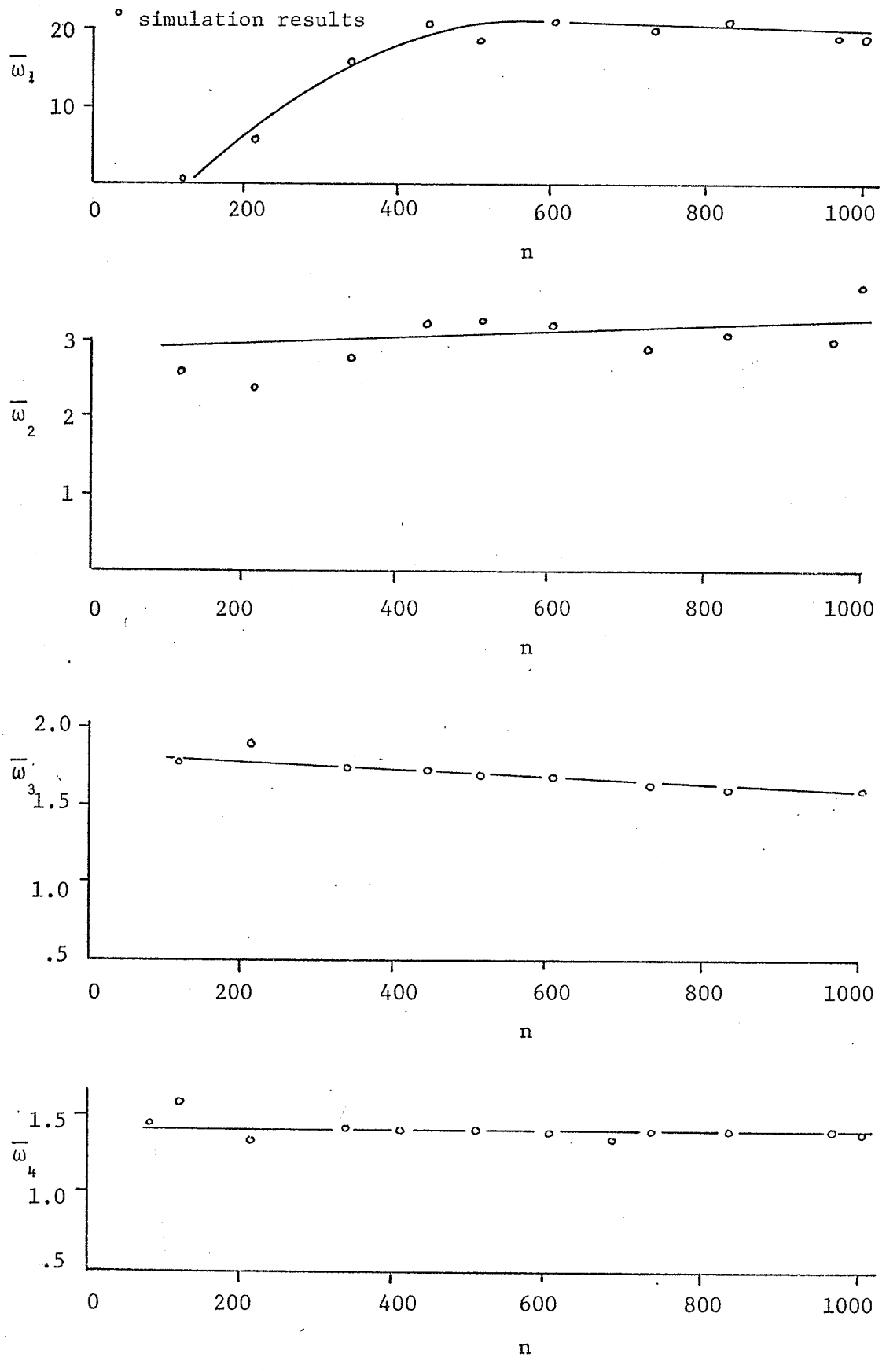
The input job-mix and operational parameters for this scheme are shown in Tables 3.2 and 3.3 and the results of each simulation for the values of C_1 are given in Appendix F.

As a further compromise between machine time and the number of jobs processed during each run, each calculation was continued to a maximum of 1000 jobs or until the mean relative response, $\overline{\omega}_j$, for each compute range j was observed to be relatively stable. These are tabulated in Appendix E. Figure 3.2 illustrates graphically the variation of the mean relative response for each compute range versus the number of jobs processed at $C_1 = 0.6$. The total number of jobs processed, n, during each of the six runs are distributed in the j compute ranges (j = 1, 2, 3, 4) and have values of 1000, 1000, 1000, 835, 1000, and 885.

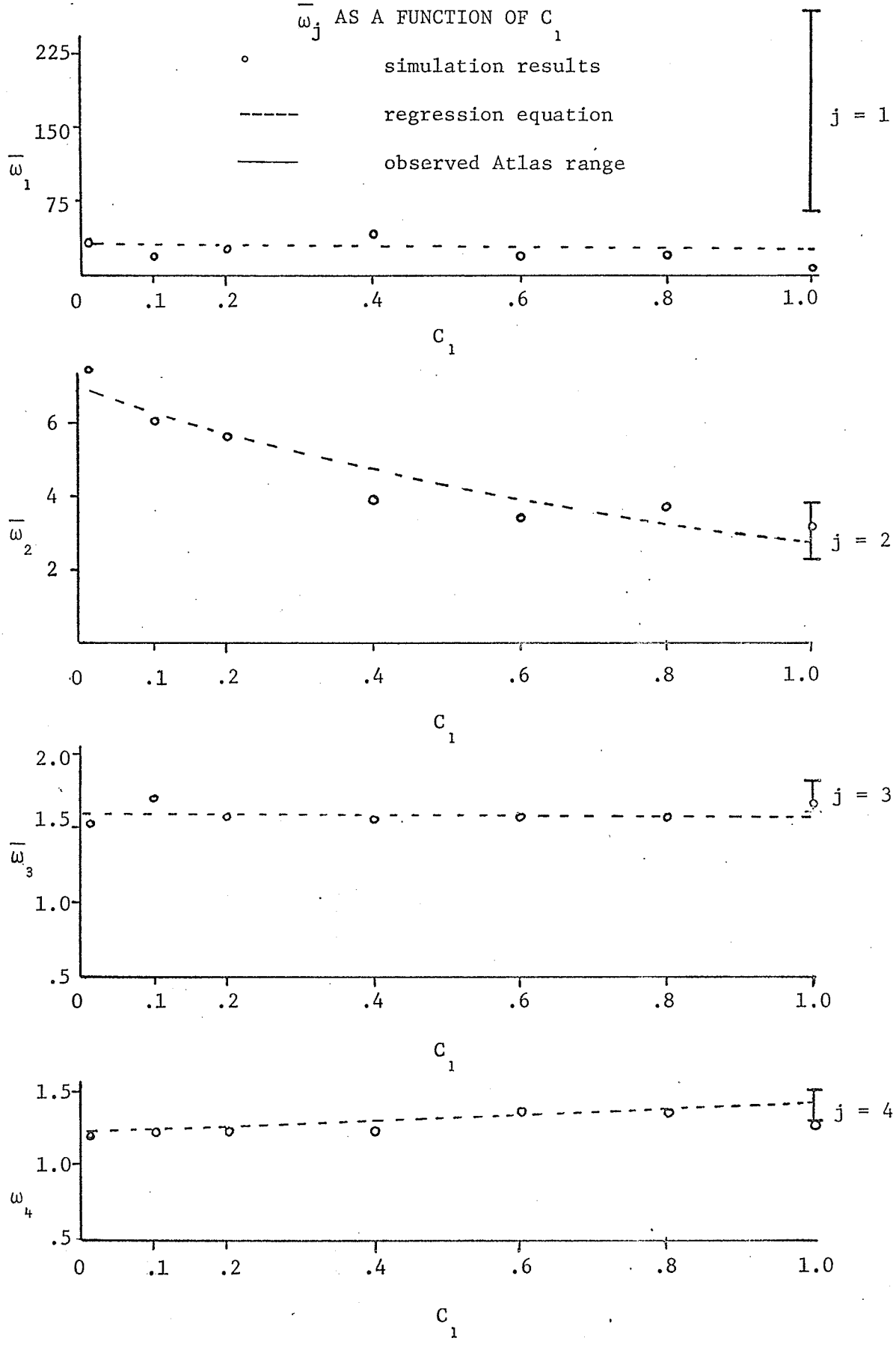
The mean relative response, taken over the n_j simulated jobs for each compute range j is denoted by $\overline{\omega}_j$. These values were calculated for each run from the job statistics gathered by the simulation model for each job i. Figure 3.3 illustrates the variation of the mean rela-

FIGURE 3.2

SKETCH OF $\overline{\omega}_j$ AS A FUNCTION OF n, $C_1 = .6$



F I G U R E 3 . 3


$\bar{\omega}_j$ AS A FUNCTION OF C_1
simulation results
regression equation
observed Atlas range
j = 1
j = 2
j = 3
j = 4
$\bar{\omega}_1$
$\bar{\omega}_2$
$\bar{\omega}_3$
ω_4
225
150
75
6
4
2
2.0
1.5
1.0
.5
0
.1
.2
.4
.6
.8
1.0
C_1

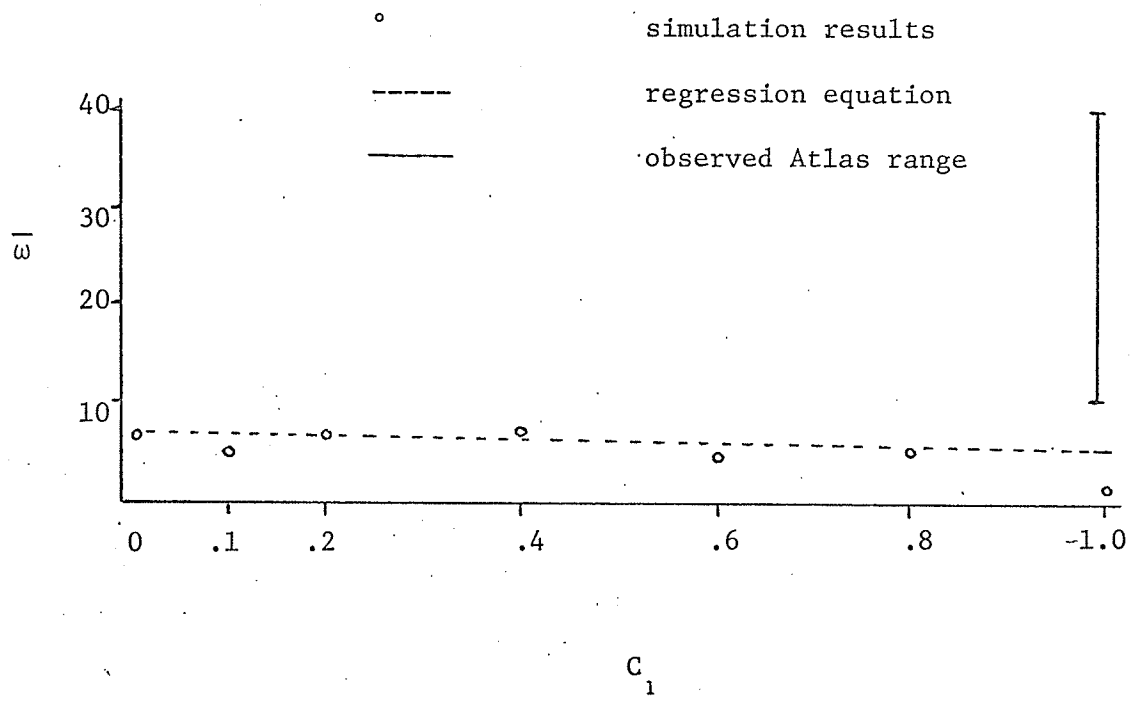
tive response, $\bar{\omega}_j$, for each compute range j versus C_1. The overall mean relative response time, $\bar{\omega}$, for all of the n simulated jobs was calculated for each run and the standard deviation ω' determined. Figure 3.4 shows the variation of the mean relative response, $\bar{\omega}$, for all jobs in each simulation run plotted against C_1.

Values of $\bar{\omega}_j$ for compute ranges j = 1, 3, and 4 appear to be linearly distributed for each value of C_1. For these compute ranges, linear extrapolation to C_1 equal to unity excludes any serious consequence. However, for the values of $\bar{\omega}_2$ (of compute range 2) logarithmic extrapolation is found to be more appropriate. By least squares regression techniques, the best fit to the values of $\bar{\omega}_j$ (j = 1, 2, 3, 4) for the values of C_1 = .01, .1, .2, .4, .6 and .8 give rise to the extrapolated values of 21.89, 2.64, 1.54 and 1.40 at C_1 = 1.0. These extrapolated values represent the mean relative response for each compute range j and are the result of the simulation of the Atlas job-stream (Figure 3.3).

In order to confirm these extrapolated values, a simulation run was performed for 336 jobs under similar conditions with C_1 equal to unity. The values for $\bar{\omega}_j$ (j = 1, 2, 3 and 4) were found to be 5.02, 3.22, 1.65, and 1.25 respectively. The deviation of these values from the extrapolated values of $\bar{\omega}_j$ (j = 2, 3, and 4) is less than 22%. In comparison, both the extrapolated and the obtained (by simulation) values of the mean relative response $\bar{\omega}_j$ for compute ranges 2, 3 and 4

F I G U R E 3 . 4

$\overline{\omega}$ AS A FUNCTION OF C_1



compare favourably to the values of $\overline{\omega}_j$ observed from the Atlas statistics. For these three compute ranges, the deviation is less than 29% and is considered a reasonable degree of accuracy for this study. Table 3.4 shows the values of $\overline{\omega}_j$ for j = 1, 2, 3 and 4 obtained by extrapolation techniques, simulating 336 jobs and those observed from the Atlas statistics.

As the Atlas job-stream is by no means constant, some variations in the mean relative response within each compute range j is observed over the sets of Atlas statistics. This variation of $\overline{\omega}_j$ is greatest in the first compute range (j = 1) and the range over all six sets is as large as [65.2, 266.7]. It is of interest to compare the mean relative response for both the extrapolated and simulation values to the corresponding ranges of $\overline{\omega}_j$ in the Atlas statistics. Results obtained by extrapolation and simulation are observed to be within the Atlas range for compute ranges 2, 3, and 4. In contrast, the values obtained by both methods, for compute range 1, is considerably lower than those observed in the Atlas statistics.

The mean relative response is the result of a division of two correlated variates (ΔT_i and τ_i) and it is interesting to compare the mean duration given by the simulation model with the mean duration in the Atlas statistics. This comparison is made in Appendix G and is found to be in good agreement with the Atlas results.

Results from the simulation run performed at C_1 equal to unity for 336 jobs give rise to a CPU utility of 55.2%, magnetic tape deck

TABLE 3.4

COMPARISON OF RELATIVE RESPONSE TIMES

COMPUTE RANGE	EXTRAPOLATED $c_1 = 1$	SIMULATION $c_1 = 1$	ATLAS	RANGE OF ATLAS
1	21.89	5.02	266.7	65.2 - 266.7
2	2.64	3.22	2.5	2.3 - 3.8
3	1.54	1.65	1.6	1.6 - 1.8
4	1.40	1.25	1.3	1.3 - 1.5

(user) utility of 38.4% and a drum utility due to paging of 22.7%. These results are seen to be in fair agreement with the values 61.0, 42.5 and 23.0 per cent respectively as calculated from the Atlas statistics used in this study.

Table 3.5 gives an example of the output statistics collected for a simulation run at ($C_1 = 1$) for 336 jobs. The means collected for the output parameters for each compute range are averaged over the n_j jobs appearing in each range.

3.5 DISCUSSION OF ATLAS SIMULATION RESULTS

In view of the assumptions, approximations and available data, the simulation of the Atlas computer is considered as being fairly successful. For compute ranges 2, 3 and 4, both the extrapolated and obtained (by simulation) values for the mean relative response lie within or on the boundaries of the range of the mean relative response observed over the six sets of Atlas statistics. In addition, the system performance based on CPU utilization and magnetic tape and drum utilization correspond to within 10% of the values calculated from the Atlas statistics. The major conclusion then is that the simulation model seems to be a reasonably valid representation of the Atlas system. Because of large variations observed in the Atlas statistics, for the mean relative response of compute range one, (Figure 3.3) it appears that the rela-

TABLE 3.5

EXAMPLE SIMULATION OUTPUT STATISTICS FOR ATLAS JOB-STREAM $C_1 = 1.0$

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURATION (SEC)	RELATIVE RESPONSE TIME	WAIT TIME (SEC)
0-1	0.32	57.0	47.05	0.77	403.41	1.01	1.44	35.51	5.02	7.45
1-8	2.90	113.0	51.64	0.31	254.60	4.95	8.17	41.41	3.22	10.83
8-120	29.76	143.0	65.27	0.40	419.99	19.78	46.14	109.03	1.65	6.32
120-960	265.73	23.0	67.26	0.43	564.13	52.80	69.57	422.64	1.25	1.61

$\bar{\omega} = 2.72$ $\omega' = 7.45$

CPU UTILITY = 52.24%

DRUM UTILITY = 22.72%

MAGNETIC TAPE UTILITY = 38.34%

NUMBER JOBS PROCESSED = 336

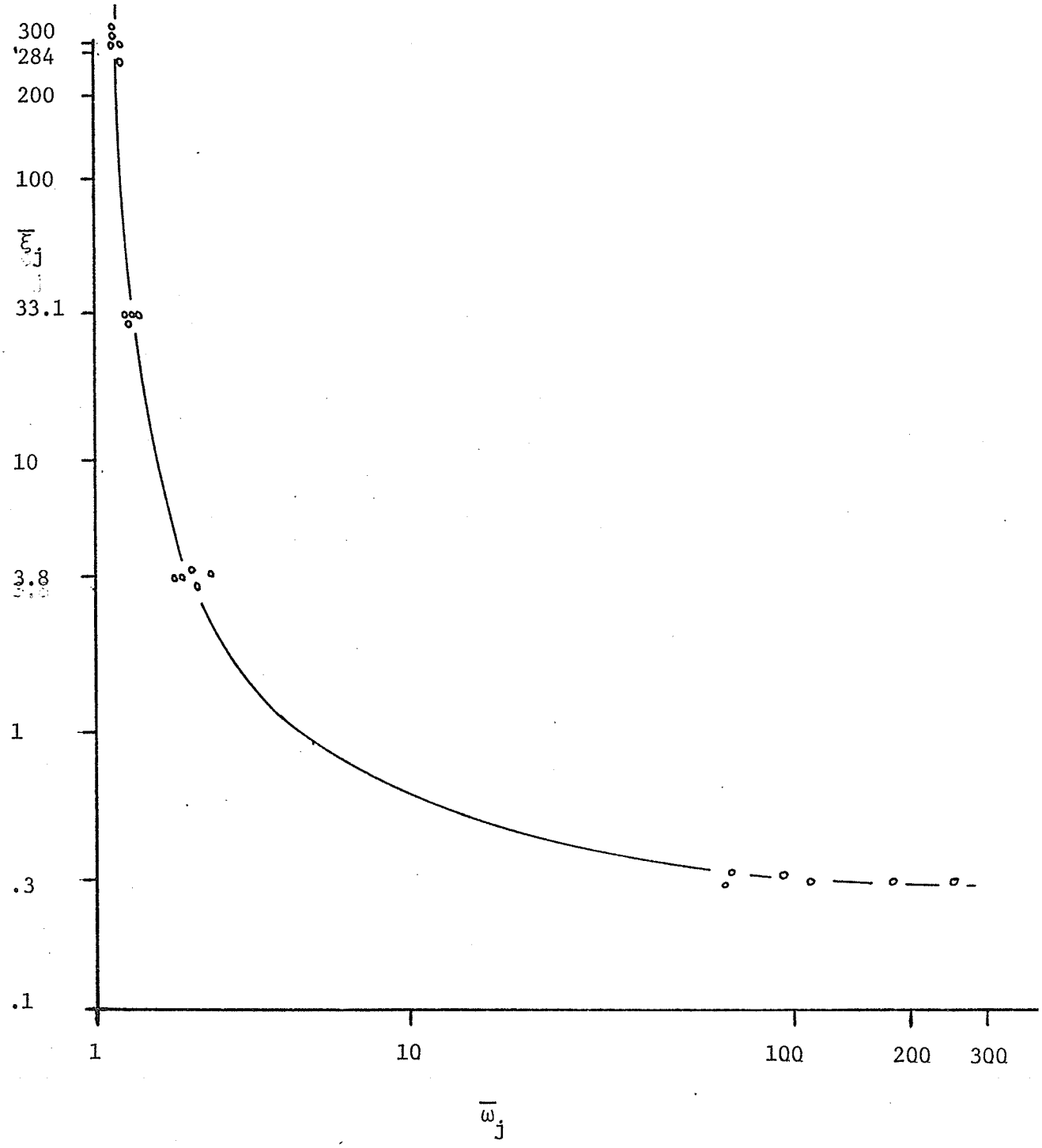
tive response for this compute range is relatively unstable. This fluctuation of the mean relative response, $\overline{\omega}_1$, may be due to either correlations in the input Atlas statistics or the streaming of jobs by the operator, operator interventions, system restarts, or all four. This conjecture is partially supported by the correlations observed in the Atlas input and output statistics. A correlation is observed between the mean relative response $\overline{\omega}_j$ (j = 1, 2, 3, 4) and the values for the mean compute and drum device times. Figure 3.5 and Figure 3.6 show graphically the correlations observed in the six sets of Atlas statistics between the mean relative response, the mean compute time and the mean drum device time respectively.

From Figure 3.5, the curve sketched in the region of points observed for the mean relative response times of compute range 1 is seen to be almost horizontal. This suggests even wider fluctuations may be expected for the mean relative response of this compute range had the Atlas statistics been collected over a longer period of time.

In the results from the simulation model large fluctuations are not observed for compute range 1. It is for this reason that a marked deviation of the mean relative response, $\overline{\omega}$, of all the n jobs simu-

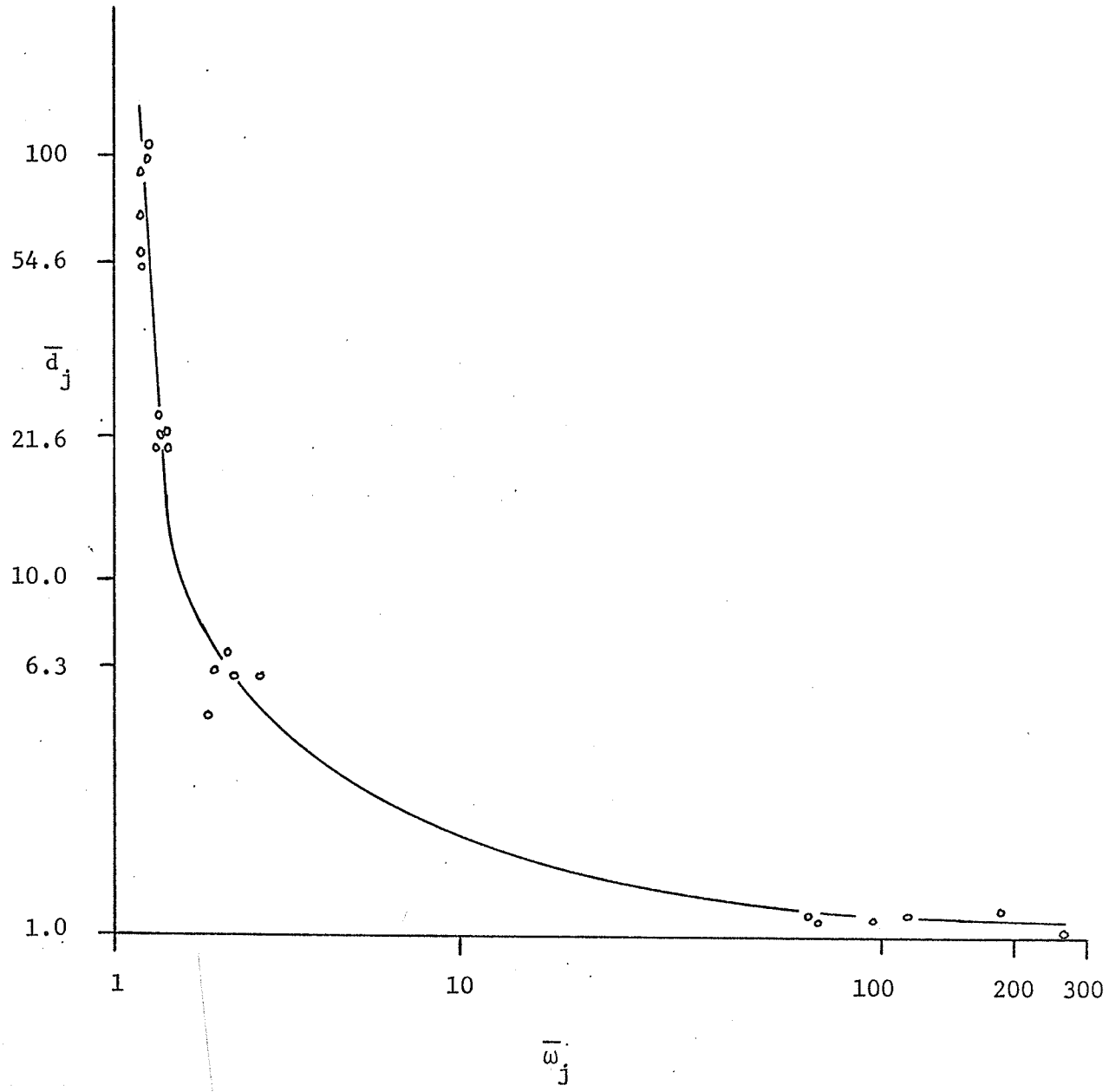
F I G U R E 3 . 5

PLOTTED VALUES OF $\overline{\xi}_j$ VERSUS $\overline{\omega}_j$ (j = 1, 2, 3, 4)



F I G U R E 3 . 6

PLOTTED VALUES OF $\overline{d}_j$ VERSUS $\overline{\omega}_j$; (j = 1, 2, 3, 4)



lated is observed in the simulation results (Figure 3.4). Because the relative responses for compute ranges 2, 3 and 4 are in good agreement with the Atlas statistics, it is reasonable to assume that the disagreement for compute range 1 is due to a feature in the operation of the Atlas system not included, or approximated to, in the simulation model, rather than due to the invalidity of the model. Possible causes for disagreement, not included in the simulation model, are:

(a) operator priorities, where priorities of jobs are assigned or altered by the computer operator over and above the system assigned priority;

(b) system restarts of the Atlas system, after which jobs remaining in the job queue are flushed to be processed at a later time;

(c) non-exponential interarrival times of jobs in the Atlas system; and

(d) neglect of a feature in the scheduling algorithm which favours jobs requiring a free peripheral output device.

One would expect the mean relative response to increase for each compute range as the system paging loadings (C_1) increase. For example, the time a job spends in the system, ΔT_i, would on the average increase by a greater amount than the increase in τ_i which depends on the CPU time, magnetic tape and drum device times. This conjecture is sup-

ported by the mean relative response observed for compute range 4. The mean relative response of compute range 1 decreases slowly for values of C_1 and the mean relative response of range 3 is almost stable. However, for each value of C_1, the mean relative response of compute range 2 is seen to decrease markedly for increasing values of C_1. This does not seem to be unreasonable if the relationship between the job type (tape or non-tape) and the allocation of the central processor is considered.

From the data used (Table 3.2), a smaller proportion of tape jobs is executed in compute range 2 which possess the lowest mean for the number of tapes. Therefore fewer jobs of this compute range are tape jobs which assume the highest priority in the simulation model. In addition, long and tape jobs are scheduled through each stage of assembly, execution and output by the simulation model in a serial manner clearing each stage before another begins. Short jobs pass through each stage independently of these. For combinations of short and long jobs on the execution list, the interval of time the CPU is allocated and awaits allocation is dependent on the rate of page requests for jobs on the execution list. For example, if the execution list is shared between these two types of jobs, the long job could seize the CPU until interrupted by a page transfer request, before the short job received any attention from the central processor. If possible, the CPU is allocated to another job as a page transfer proceeds. As a result, the distribution of attention given to

jobs on the execution list by the CPU becomes more uniform as the paging rate approaches some limited value. Thus, it is postulated that the effect of more frequent CPU switching between jobs due to heavier paging demands is reflected in the simulation results by the decreasing mean relative response, $\overline{\omega}_j$, of compute ranges 1 and 2 for increasing values of C_1.

CHAPTER IV

SIMULATION OF ATLAS SYSTEM USING DEMAND FUNCTION $f_j(p)$

4.1 INTRODUCTION

In the simulation of the Atlas system described in Chapters II and III some simplifying assumptions and approximations were adopted. For example, each job within a particular compute range assumes the same mean interarrival time for out-of-core requests which are independent of time. The interarrival of out-of-core page requests is difficult to estimate since during execution of a job the fraction of the total number of pages resident in core is variable. It is apparent then that core storage is shared by the jobs concurrently on the execution list. A relationship between the amount of execution storage allocated to a job and the average length of the interval of time a job may execute uninterrupted has been suggested [20].

In the next section a similar relation is described and will be referred to as the demand function. This function was used to generate the mean interarrival times ε_{ij} for each job i for compute range

j. For each compute range a particular demand function is defined relating the mean interarrival time of out-of-core page requests of job i to the proportion of pages found in core storage.

It is expected that if the storage allocated is relatively small the length of an execution burst is short. In contrast, if the entire job is maintained in core storage, the mean execution interval is dependent only on other interrupts suspending execution.

The required paging operations during execution give rise to additional supervisor overheads and increases the job halt time while a job awaits completion of page transfers. In reference to the Atlas computer, the logged output statistics (Table 3.1) indicate that the mean swap time required for paging operations exceed the mean compute time by a factor greater than 1 for jobs of compute ranges 1 and 2. This high ratio of swap time to compute time is suspected to be a result of the limited amount of core storage available and the large amount of swapping during compilation.

It has been suggested [20] that the fundamental problem encountered by multiprogramming involving paging techniques is the conflict between the number of jobs that may be multiprogrammed and the system's ability to honour all storage demands. In the Atlas computer and simulation model the maximum number of jobs multiprogrammed is two.

The simulation model described in Chapter II was modified slightly to include the demand function to generate the mean interarrival time

for out-of-core page requests. The effect of this function is demonstrated using the same job-mix parameters used in the Atlas simulation described in the previous Chapter.

4.2 THE DEMAND FUNCTION

In this section an analytical relationship between the mean interarrival time of out-of-core page requests and the proportion of in-core pages for jobs of each compute range is determined on the basis of a previous publication by Belady [20]. Some of the underlying assumptions made for his analytical approach were relaxed. The assumptions retained for this simulation study are:

(a) Jobs are executed in a paging environment in which the storage demands usually exceed the available core storage capacity.

(b) The average time interval of uninterrupted processing is a function of the number of pages resident in core storage.

(c) The pages are referenced during execution in a 'random' sequence.

The relationship or demand function determined for the simulation model for each compute range j is defined to be:

$$f_j(p) = k_j p/(1-p) \quad , \qquad .....(4.1)$$

where k_j is a constant and p the proportion of the total pages R_{ij} a job i has resident in core storage. The constant k_j was taken to be

unity by Belady's approximation since the average length of an uninterrupted execution interval was considered to be a function of only the storage space occupied by each job. However, for the function adopted in this study the value of k_j for each compute range j is such that the average execution interval corresponds to the value calculated from the Atlas statistics (equation 2.2.9).

It is assumed that for two jobs on the Atlas execution list, the mean number of pages resident in core is 16. Thus, the mean proportion of in-core pages for jobs of compute range j is:

$$\bar{p}_j = 16/\bar{R}_j \quad , \qquad .....(4.2)$$

where $\bar{R}_j$ is the mean storage blocks requested for the job within compute range j. The observed mean interarrival time, $\bar{\varepsilon}_j$, for jobs of compute range j can therefore be expressed:

$$\bar{\varepsilon}_j = f_j(\bar{p}_j) \quad . \qquad .....(4.3)$$

Using equation 4.1 the constant k_j for each compute range j is determined as:

$$k_j = \bar{\varepsilon}_j(1 - \bar{p}_j)/\bar{p}_j \quad . \qquad .....(4.4)$$

The interarrival times ε_{ij} for an out-of-core page request for each job i are assumed to be normally distributed about $f_j(p)$. When the proportion of in-core pages is p_0 the mean of the distribution is given by $f_j(p_0)$ and the standard deviation taken to be $f_j(p_0)/6$.

An interarrival time, ε_0^*, of the next out-of-core page request generated at time t_0 is subject to change as the proportion p_0 changes

at a later time t_i to p_i (i=1, 2, 3, ...1) during the course of execution. The interarrival time ε_i^* after the i^{th} change in proportion p_0 may be defined by the recursive relationship:

$$\varepsilon_i^* = \frac{\varepsilon_0^*}{f_j(p_0)} f_j(p_i)\{1 - \frac{t_i - t_{i-1}}{\varepsilon_{i-1}^*}\} \quad , \qquad .....(4.5)$$

such that the ratio $(t_i - t_{i-1})/\varepsilon_{i-1}^* < 1$ for the request not to have already occurred.

Thus, ε_i^* is the CPU time necessary to give rise to the next page transfer after the proportion of in-core pages has changed for the i^{th} time and ε_0^* is the initial interarrival time generated from the distribution with mean $f_j(p_0)$. A detailed derivation of the recursive relation (equation 4.5) is given in Appendix H.

4.3 INVESTIGATION OF THE RELIABILITY OF THE SIMULATION MODEL USING DEMAND FUNCTION

The use of the demand function requires that the model calculates the interarrival time of the next out-of-core page request for a job i each time its proportion of in-core pages is changed. It is assumed that a job is loaded into core storage when a job is entered on the execution list.

In order to examine the reliability of the simulation results using this demand function, n jobs were processed through the simulation model

and the simulation output statistics calculated for continued values of n equal to 100, 165, 266, 386, 481, 641, 702, 766 and 1000. The job-mix and operational parameters used are similar to those described in Chapter II and are given in Tables 4.1 and 4.2 respectively.

The mean drum device time, $\overline{d}_j$, obtained for each compute range j is output by the simulation model and is compared to the range of the corresponding value given in the Atlas statistics. For increasing values of n the simulation results for $\overline{d}_j$ are plotted against n as shown in Figure 4.1. The value of $\overline{d}_j$ (j = 1, 2, 3 and 4) obtained from the 1000 simulated jobs are 1.28, 5.90, 16.65 and 46.92. These values correspond reasonably well to the Atlas results which are observed to be 1.0, 6.3, 21.6 and 54.7. The largest deviation (j = 3) is less than 22%. Table 4.3 gives a comparison of $\overline{d}_j$ obtained by simulation involving the demand function $f_j(p)$ and the observed Atlas results.

The mean duration, $\overline{\Delta T}_j$, of jobs within each compute range given in the Atlas statistics and output by the simulation model is used as a further measure of the reliability of the simulation results. For the n continued values given above the values obtained for $\overline{\Delta T}_j$ are graphically presented in Figure 4.2 and the values obtained at n equal to 1000 are compared to the range of the corresponding Atlas results in Table 4.4. The largest deviation of the simulation results (j = 2) is less than 29%. However, the values obtained for $\overline{\Delta T}_j$ (j = 1, 2, 3, and 4) are observed to lie within or near the boundaries of each

TABLE 4.1

INPUT JOB-MIX PARAMETERS USED FOR RELIABILITY STUDY WITH DEMAND FUNCTION $f_j(p)$

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS-FERRED (BLKS)	PRINTER OUTPUT (BLKS)
0 - 1	0.3	360.0	48.3	0.8	423.6	1.4
1 - 8	3.8	864.0	51.3	0.4	344.2	8.3
8 - 120	33.1	1333.0	66.5	0.5	449.6	45.2
120 - 960	284.0	194.0	65.8	0.5	452.5	70.8

TABLE 4.2

OPERATION PARAMETERS USED IN RELIABILITY STUDY WITH DEMAND FUNCTION $f_j(p)$

Number of generation slots,	$n_g = 4$
Number of execution slots,	$n_e = 2$
Number of assembly queue slots,	$n_q = 20$
Number of drum queue slots,	$n_d = 3$
Number of tape queue slots,	$n_t = 2$
Drum response time (sec),	$d_r = .014$
Tape response time (sec),	$t_r = .063$
Size of core storage,	$C = 32$
Size of drum storage,	$D = 133$
Number of user magnetic tapes,	$mt = 5$
(Central executive) supervisor overhead (sec),	$t_{s_1} = .006$
(Switching) supervisor overhead (sec),	$t_{s_2} = .002$
Proportionality constants	$k_1 = 0.017$
	$k_2 = 0.037$
	$k_3 = 0.135$
	$k_4 = 0.453$
Starting random number	= 568976679

FIGURE 4.1

$\overline{d}_j$ AS A FUNCTION OF n

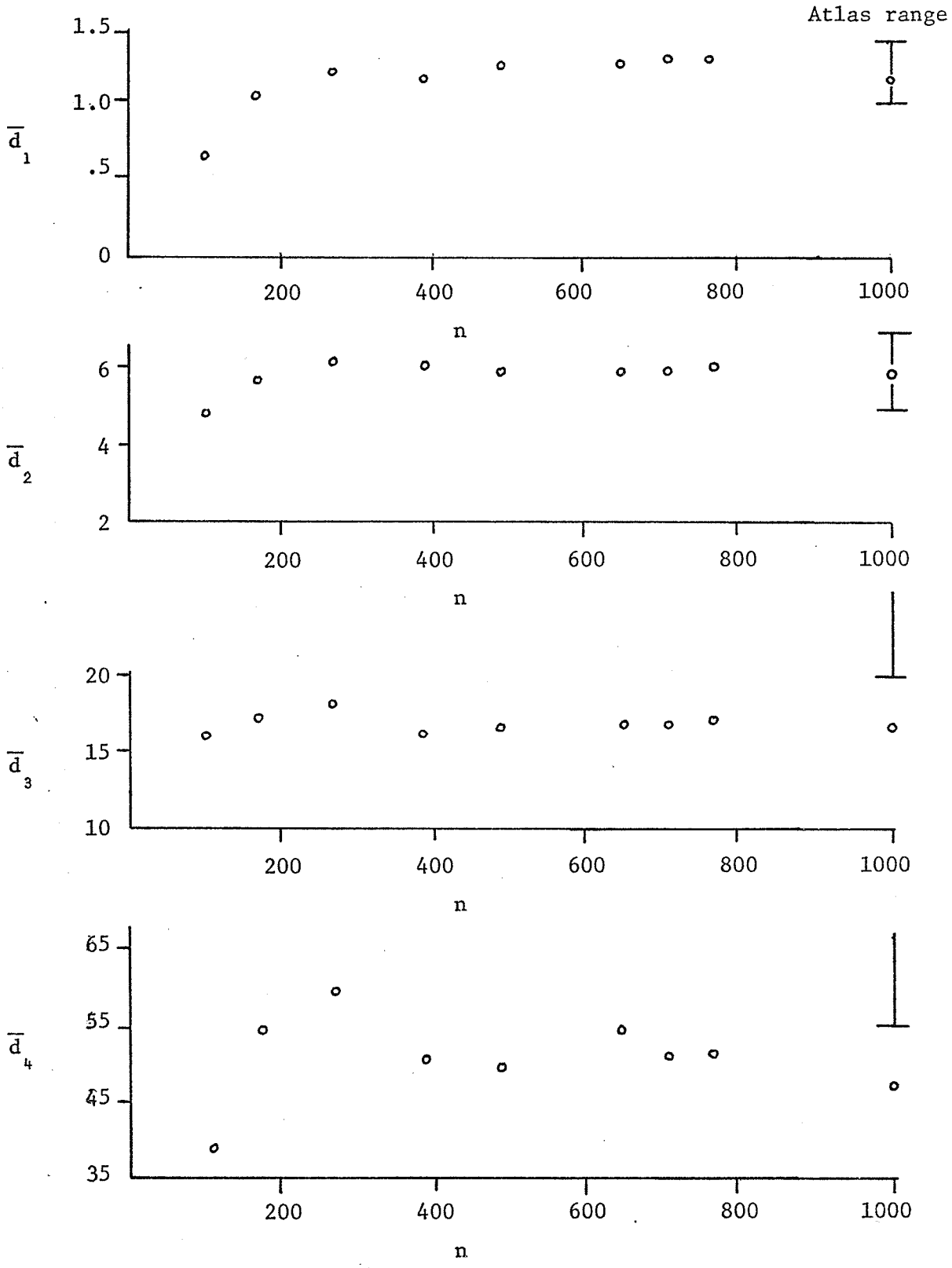


TABLE 4.3

COMPARISON OF $\bar{d}_j$ FROM SIMULATION USING $f_j(p)$

	SIMULATION	ATLAS	ATLAS RANGE
$\bar{d}_1$	1.28	1.0	1.0 - 1.4
$\bar{d}_2$	5.90	6.3	4.9 - 6.9
$\bar{d}_3$	16.65	21.6	20.7 - 24.7
$\bar{d}_4$	46.92	54.7	54.7 - 105.2

FIGURE 4.2

$\overline{\Delta T}_j$ AS A FUNCTION OF n

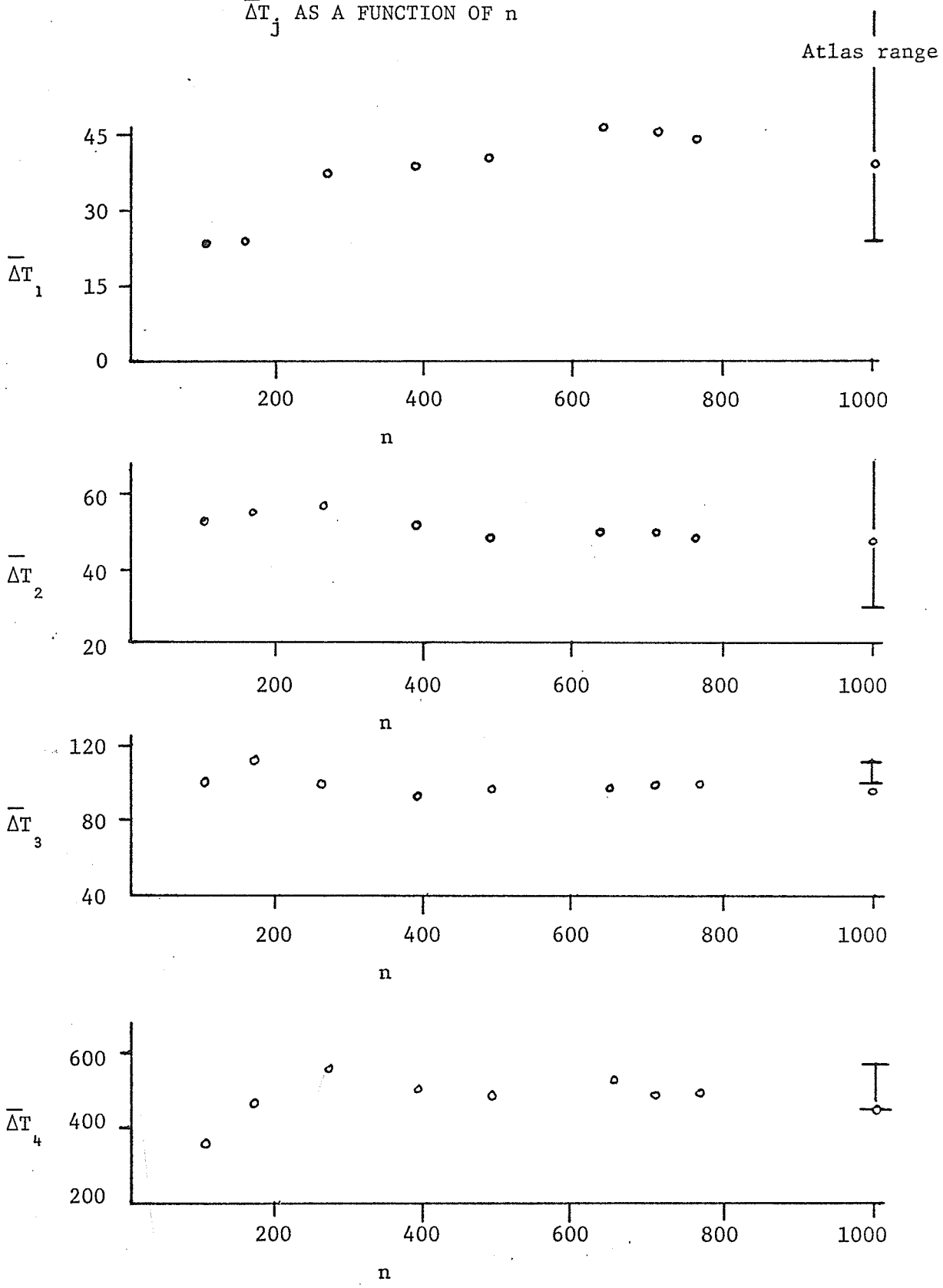


TABLE 4.4

COMPARISON OF $\overline{\Delta T}_j$ FROM SIMULATION USING $f_j(p)$

	SIMULATION	ATLAS	ATLAS RANGE
$\overline{\Delta T}_1$	38.59	80.8	26.1 - 80.8
$\overline{\Delta T}_2$	48.46	37.7	30.0 - 45.9
$\overline{\Delta T}_3$	97.25	102.2	102.2 - 113.2
$\overline{\Delta T}_4$	455.35	456.9	456.9 - 569.4

range of $\overline{\Delta T}_j$ observed over the six sets of Atlas data (Table 3.1).

As expected the mean relative response $\overline{\omega}_j$, (j = 2, 3 and 4), which involves (see equation 2.5.7) both of the discussed variables (ΔT_j and d_i) is also found to be in fairly good agreement with the Atlas values. The results from the simulation for $\overline{\omega}_j$ are 23.00, 3.08, 1.43 and 1.28 and the corresponding Atlas values were observed to be 266.7, 2.5, 1.6 and 1.3. The greatest deviation of $\overline{\omega}_j$ (where j = 2, 3 and 4) from the Atlas results is less than 24% and lies within or near the boundries of the range $\overline{\omega}_j$ observed over the Atlas statistics. Figure 4.3 illustrates the plotted values of $\overline{\omega}_j$ (j = 1, 2, 3 and 4) versus n, the number of jobs processed. Table 4.5 gives a comparison of the obtained values of $\overline{\omega}_j$ when 1000 jobs are processed to those observed in the Atlas statistics.

The values obtained for the CPU, drum and magnetic tape utility are 58.12, 21.05 and 38.95 per cent respectively. Again, these values are in good agreement with the corresponding values of 61.0, 23.0 and 42.5 per cent determined from the Atlas statistics. The largest deviation of these simulation results from the Atlas values is less than 9%.

The results of the simulation using the demand function are given in Table 4.6 and compare favourably to the results of the simulation of Chapter III (the results are given in Appendix F).

4.4 DISCUSSION OF SIMULATION RESULTS INVOLVING $f_j(p)$

In additon to the simplifying assumption and the approximations described in Chapter III the adopted demand function $f_j(p)$ is a further

FIGURE 4.3

$\bar{\omega}_j$ AS A FUNCTION OF n

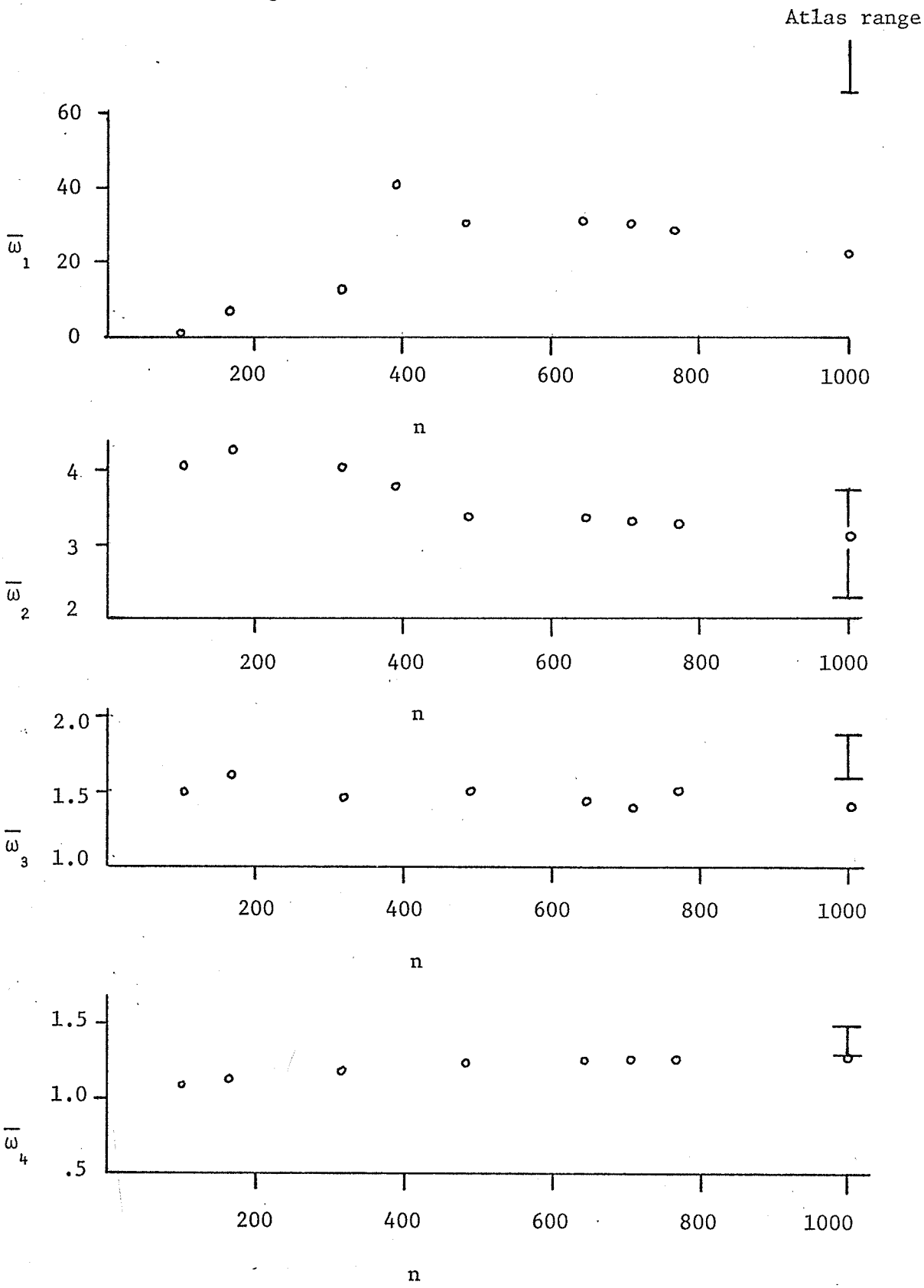


TABLE 4.5

COMPARISON OF $\overline{\omega}_j$ FROM SIMULATION USING $f_j(p)$

	SIMULATION	ATLAS	ATLAS RANGE
$\overline{\omega}_1$	23.00	266.7	65.2 - 266.7
$\overline{\omega}_2$	3.08	2.5	2.3 - 3.8
$\overline{\omega}_3$	1.43	1.6	1.6 - 1.8
$\overline{\omega}_4$	1.28	1.3	1.3 - 1.5

TABLE 4.6 OUTPUT FROM SIMULATION USING $f_j(p)$

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURATION (SEC)	RELATIVE RESPONSE TIME	QUEUE WAIT TIME
0 - 1	0.26	156.0	50.27	0.71	413.31	1.28	1.40	38.59	23.00	9.35
1 - 8	3.16	373.0	52.54	0.34	317.42	5.90	8.39	48.46	3.08	10.04
8 - 120	31.42	397.0	67.39	0.41	370.85	16.65	45.61	97.25	1.43	2.28
120 - 960	280.76	74.0	66.31	0.41	494.62	46.92	71.01	455.35	1.28	5.26

$\overline{\omega} = 5.40$ $\omega' = 40.77$

CPU UTILITY = 58.12%

DRUM UTILITY = 21.05%

TAPE UTILITY = 38.95%

NUMBER JOBS PROCESSED = 1000

approximation which is believed to be a more accurate representation of the real situation than the corresponding approximation used in Chapter III.

The agreement between the Atlas results and the simulated values using the demand function is superior to the agreement obtained in Chapter III for $\overline{\Delta T}_j$ and inferior for the $\overline{\omega}_j$ values.

It is perhaps significant to note, however, that although the results obtained in this Chapter stabilized fairly well as n increased to 1000, it is a single calculation and the results are subject to some uncertainty. More attention was paid to the reproducibility of the results in Chapter III and it is possible that were the same attention given to reproducibility in this section, the results might be superior to the ones observed.

The obtained values of the mean drum device time $\overline{d}_j$ are less than the values of the Atlas statistics for compute ranges 2, 3 and 4. This may be due to the assumptions that for each job of each compute range j, the mean number of in-core pages is taken to be 16. Secondly, mean required storage, $\overline{R}_j$ is approximated by the average of the requested compile and execution store as described in Chapter II, section 2.2.2.

4.5 CONCLUSIONS

Acknowledging the imperfections in the design of the simulation model, it is felt that simulation results obtained (as described in Chapter III and Chapter IV) from both models are in general in fairly

good agreement with the Atlas statistics collected for this study.

In spite of the simplification and approximations made in the model the output results have indicated that the simulation model behaved much like the actual Atlas computer.

It is apparent that in a moderately complex computer system such as the Atlas several aspects or features contribute more than others to the overall performance of the computer. This simulation study has indicated that paging computer systems of this nature may successfully be simulated by a next event type simulation model if the more influential features can be isolated and replicated in the model. The less important features may be simplified or excluded without serious consequences. For example, the assumption that overheads of less than 1 millisecond may be ignored without altering the overall operation of the model is in actual fact quite drastic.

APPENDIX A

DATA MAINTAINED ON JOBS BY SIMULATION MODEL DURING PROCESSING

The job description for each entry on the job list includes:

1. program number;
2. compute range (identifier);
3. time of arrival;
4. CPU time requested;
5. storage (blocks) requested;
6. number of user tapes requested;
7. number of blocks of output requested.

The additional descriptions recorded for an entry onto the execution list include;

8. time assembly terminated (TIME ON);
9. time loaded into core;
10. accumulated CPU time received;
11. accumulated drum time received;
12. accumulated tape time received.

A P P E N D I X B

SAMPLE OF GENERATED JOB-STREAM

A sample, consisting of the first few jobs appearing in each compute range of the job stream is given below. The job-mix and operational parameters used to generate this job stream are given in Tables 3.2 and 3.3 respectively.

COMPUTE RANGE ONE

	PROGRAM NUMBER	TIME ON	CPU TIME REQUEST (SEC)	STORAGE BLOCKS	MAGNETIC TAPES	BLOCKS OUTPUT
1	1	123.13	0.62	33	1	1
2	6	489.52	0.07	55	2	1
3	16	981.15	0.06	55	0	2
4	23	1105.40	0.27	68	1	1
5	28	1368.50	0.19	35	0	2
6	34	2252.84	0.56	61	1	2
7	44	2349.15	0.09	35	0	1
8	51	2959.88	0.36	49	0	1
9	60	3152.30	0.11	36	1	2
10	63	4248.68	0.28	51	1	1

COMPUTE RANGE TWO

1	2	127.37	5.17	55	0	11
2	7	259.61	3.78	48	0	6
3	9	262.15	5.26	33	0	5
4	10	404.44	2.79	34	0	10
5	13	421.54	1.03	61	0	10
6	14	483.44	1.16	57	1	10
7	15	551.63	1.03	59	0	4
8	17	622.72	6.72	64	0	5
9	18	986.19	1.43	63	1	8
10	21	1025.95	1.67	73	0	10

COMPUTE RANGE THREE

	PROGRAM NUMBER	TIME ON	CPU TIME REQUEST (SEC)	STORAGE BLOCKS	MAGNETIC TAPES	BLOCKS OUTPUT
1	3	34.88	57.42	58	1	40
2	5	236.21	27.22	57	1	22
3	8	271.42	21.07	52	1	14
4	11	377.96	18.42	59	0	50
5	12	655.15	109.16	85	2	52
6	19	714.42	42.75	62	1	50
7	20	1006.12	18.99	71	1	31
8	22	1034.03	19.68	58	0	66
9	25	1105.40	8.32	97	0	37
10	26	1297.48	9.46	87	0	33

COMPUTE RANGE FOUR

	PROGRAM NUMBER	TIME ON	CPU TIME REQUEST (SEC)	STORAGE BLOCKS	MAGNETIC TAPES	BLOCKS OUTPUT
1	4	1147.70	149.51	49	0	64
2	27	1326.00	173.04	91	0	56
3	30	1851.32	325.13	60	0	80
4	41	2452.22	151.85	74	2	52
5	52	3416.08	300.33	84	2	71
6	69	3599.34	274.93	41	1	85
7	70	6452.23	171.47	108	0	87
8	103	7646.44	425.09	25	0	64
9	129	8228.73	266.99	20	0	78

APPENDIX C

DERIVATION OF THE MEAN FOR THE MAGNETIC TAPE DISTRIBUTIONS

In Chapter II, an exponential distribution was assumed to generate variates (rounded to the nearest integer value) which represent the number of user magnetic tapes requested by a job i. This integer distribution is of a geometric type and the actual mean $\overline{m}$ determined as:

$$\overline{m} = \frac{\sum_{j=1}^{5} \int_{j-0.5}^{j+0.5} \frac{1}{\overline{U}_j} e^{-U_{ij}/\overline{U}_j} \, dU_{ij}}{\int_{0}^{5.5} \frac{1}{U_j} e^{-U_{ij}/\overline{U}_j} \, dU_{ij}} \quad \ldots\ldots(C.1)$$

For the values of $\overline{U}_j$ used throughout the calculations (Table 3.2);

$$\int_{0}^{5.5} \frac{1}{\overline{U}_j} e^{-U_{ij}/\overline{U}_j} \, dU_{ij} \simeq 1 \quad .$$

Using this approximation and denoting $e^{-1/\overline{U}_j}$ by σ equation C.1 reduces to upon integration;

$$\overline{m} = \sigma^{1/2}\{1 + \sigma + \sigma^2 + \sigma^3 + ...\} \quad ,$$

hence,

$$\overline{m} = \sigma^{1/2}/(1 - \sigma) \quad . \qquad .....(C.2)$$

A P P E N D I X D

SAMPLE CPU ALLOCATION PROFILES

In Chapter II the allocation of the CPU is described. Three partial profiles depicting typical situations encountered during multiprogramming as in the Atlas computer are shown.

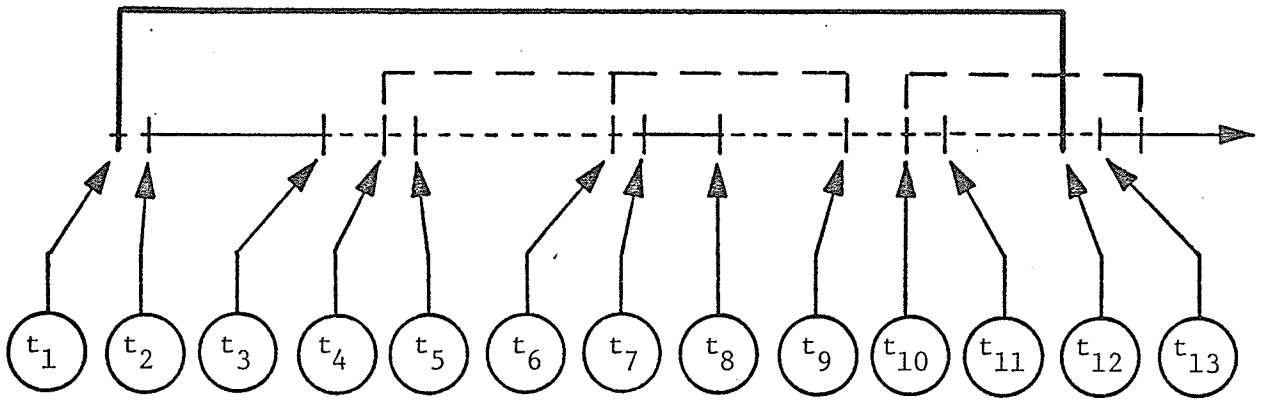
(1) Figure D.1 illustrates the sequence of events as the execution list contains a tape and a short job. The tape job is engaged in a tape transfer and control is passed to the short job.

(2) The situation illustrated by Figure D.2 resulted from the execution list containing a tape and a short job. The tape job (having highest priority) is engaged in a drum transfer. Allocation of the CPU to the short job is inhibited in this situation as shown.

(3) The last profile, Figure D.3, reflects the execution list containing a long and short job of equal priority. As a job is halted during a drum transfer, the CPU is allocated if possible to the other job on the execution list. In this situation, the CPU is allocated on a first-found-first-served basis, with no decision as to priority taken in account.

FIGURE D.1

PROFILE OF TAPE AND SHORT JOB - SITUATION 1

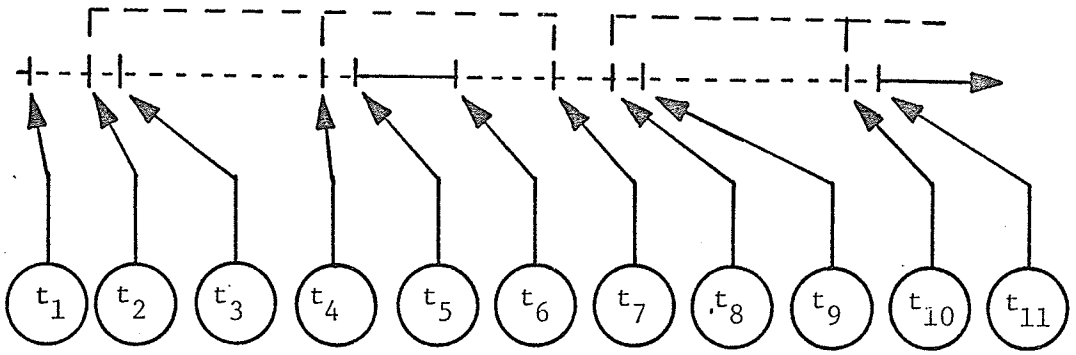


———	tape transfer
— — — —	drum transfer
- - - - - - - -	cpu idle or overheads
———	cpu executing

t_1 = 66.439;	start tape transfer
t_2 = 66.441;	start short job executing
t_3 = 66.450;	request page from drum for short job
t_4 = 66.456;	start page transfer into core
t_5 = 66.458;	end of supervisor cycle
t_6 = 66.470;	page in core, start page transfer to drum
t_7 = 66.472;	start short job executing
t_8 = 66.477;	request page from drum, queue request
t_9 = 66.485;	page on drum, reapply for queued request
t_{10}= 66.490;	start page transfer into core
t_{11}= 66.492;	end of supervisor cycle
t_{12}= 66.502;	end of tape transfer
t_{13}= 66.504;	start tape job executing

FIGURE D.2

PROFILE OF TAPE AND SHORT JOB - SITUATION 2



— — — — drum transfer

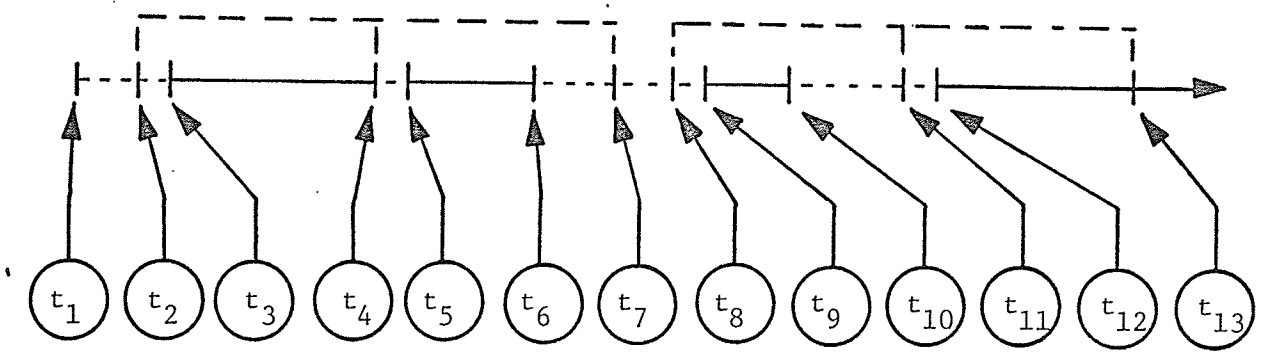
- - - - - - - - cpu idle or overheads

———— cpu in execution

t_1 = 45.052; request page from drum for tape job
t_2 = 45.058; start page transfer into core
t_3 = 45.060; end of supervisor cycle
t_4 = 45.072; page in core, start page transfer to drum
t_5 = 45.074; start tape job executing
t_6 = 45.081; request page from drum, queue request
t_7 = 45.088; page on drum, reapply for queued page request
t_8 = 45.094; start page transfer into core
t_9 = 45.096; end of supervisor cycle
t_{10} = 45.108; page in core, start page transfer to drum
t_{11} = 45.110; start tape job executing

FIGURE D.3

PROFILE OF LONG AND SHORT JOB - SITUATION 3



— — — — drum transfer

- - - - - - - - - cpu idle or overheads

——————— cpu execution

t_1 = 53.270; request page from drum for short job

t_2 = 53.276; start page transfer into core

t_3 = 53.278; start long job executing

t_4 = 53.290; page in core, start page transfer to drum

t_5 = 53.292; start short job executing (first found first served)

t_6 = 53.297; request page from drum, queue request

t_7 = 53.304; page on drum, reapply for queued page request

t_8 = 53.310; start page transfer into core

t_9 = 53.312; start long job executing

t_{10} = 53.319; request page from drum, queue request

t_{11} = 53.324; page in core, start page transfer to drum

t_{12} = 53.326; start short job executing

t_{13} = 53.338; page on drum, reapply for queued page request

APPENDIX E

DISTRIBUTION OF $\overline{\omega}_j$ AS A FUNCTION OF n

The following represents the tabulated values of the mean relative response for each compute range as a function of the number of jobs processed, n, for C_1 = .01, .1, .2, .4, .6, .8, and 1.0.

	n	$\overline{\omega}_1$	$\overline{\omega}_2$	$\overline{\omega}_3$	$\overline{\omega}_4$
C_1 = .01	1000	31.69	7.55	1.52	1.20
C_1 = .1	163	2.53	1.55	1.23	1.09
	463	4.17	5.99	1.57	1.18
	881	19.69	6.36	1.61	1.24
	1000	18.87	6.04	1.70	1.23
C_1 = .2	235	60.27	10.36	1.58	1.36
	528	31.39	5.84	1.34	1.29
	759	27.23	5.36	1.50	1.30
	937	27.60	5.71	1.62	1.28
	1000	25.94	5.74	1.59	1.28
C_1 = .4	135	37.95	3.81	1.57	1.07
	271	26.20	3.25	1.56	1.05
	364	31.12	4.03	1.63	1.17
	455	45.98	3.95	1.58	1.24
	565	37.08	3.93	1.50	1.20
	657	45.51	3.72	1.54	1.21
	748	44.83	4.01	1.51	1.23
	835	41.10	3.99	1.55	1.24
C_1 = .6	117	1.30	2.53	1.75	1.56
	203	5.98	2.33	1.89	1.38
	337	15.28	2.79	1.74	1.41
	430	20.19	3.19	1.71	1.40
	595	20.82	3.13	1.67	1.38
	727	19.50	2.87	1.60	1.37
	820	21.23	3.08	1.59	1.38
	958	18.80	2.59	1.55	1.39
	1000	18.88	3.65	1.57	1.37

	n	$\bar{\omega}_1$	$\bar{\omega}_2$	$\bar{\omega}_3$	$\bar{\omega}_4$
$C_1 = .8$	131	23.71	4.48	1.78	1.58
	227	32.90	4.71	1.60	1.47
	316	26.89	4.24	1.50	1.36
	400	27.02	5.11	1.55	1.51
	488	38.55	4.60	1.50	1.48
	591	31.64	4.08	1.48	1.44
	705	25.62	4.08	1.56	1.38
	790	23.79	3.87	1.55	1.38
	885	21.55	3.73	1.55	1.36
$C_1 = 1.0$	92	3.98	4.24	1.29	1.31
	180	5.69	3.60	1.44	1.19
	241	4.65	3.12	1.39	1.20
	336	5.02	3.22	1.65	1.25

APPENDIX F

SIMULATION OUTPUT STATISTICS FOR ATLAS JOB-STREAM C_1 = .01

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURATION (SEC)	RELATIVE RESPONSE TIME	WAIT TIME (SEC)
0-1	0.26	145.0	45.61	0.63	309.13	0.01	1.38	33.51	31.69	7.11
1-8	3.15	338.0	51.68	0.30	291.83	0.03	8.22	39.14	7.55	6.36
8-120	31.29	436.0	66.12	0.44	450.54	0.11	45.76	79.36	1.52	2.81
120-960	294.54	81.0	67.98	0.43	477.23	0.30	69.67	393.00	1.20	0.35

$\bar{\omega}$ = 7.91 ω' = 53.28

CPU UTILITY = 64.75%

DRUM UTILITY = .14%

MAGNETIC TAPE UTILITY = 40.00%

NUMBER JOBS PROCESSED = 1,000

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURATION (SEC)	RELATIVE RESPONSE TIME	WAIT TIME (SEC)
0-1	0.26	130.0	48.38	0.78	416.99	0.05	1.38	32.04	18.75	4.36
1-8	3.21	356.0	50.37	0.28	234.28	0.41	8.11	38.05	6.04	10.98
8-120	33.55	439.0	66.60	0.41	421.34	1.97	45.79	88.17	1.70	5.06
120-960	286.72	75.0	65.63	0.44	486.22	5.18	67.12	394.43	1.23	0.38

$\overline{\omega}$ = 5.44 ω' = 41.16

CPU UTILITY = 62.36%

DRUM UTILITY = 2.34%

MAGNETIC TAPE UTILITY = 37.70%

NUMBER JOBS PROCESSED = 1,000

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURA-TION (SEC)	RELAT-IVE RES-PONSE TIME	WAIT TIME (SEC)
0-1	0.28	149.0	49.68	0.68	337.04	0.15	1.40	34.08	25.94	11.18
1-8	3.41	321.0	52.03	0.31	300.66	1.02	8.27	47.88	5.74	15.16
8-120	30.88	456.0	66.59	0.44	457.32	3.87	45.32	85.78	1.59	4.25
120-960	303.87	74.0	65.81	0.39	452.92	11.48	72.18	422.14	1.28	4.06

$\overline{\omega}$ = 6.53 ω' = 44.03

CPU UTILITY = 65.20%

DRUM UTILITY = 5.13%

MAGNETIC TAPE UTILITY = 42.35%

NUMBER JOBS PROCESSED = 1,000

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURATION (SEC)	RELATIVE RESPONSE TIME	WAIT TIME (SEC)
0-1	0.26	117.0	48.14	0.72	400.23	0.29	1.41	45.81	41.10	18.86
1-8	3.02	268.0	52.05	0.32	293.50	1.93	8.28	39.28	3.99	9.40
8-120	32.68	387.0	65.90	0.46	512.00	8.60	44.66	101.44	1.55	3.56
120-960	326.20	63.0	67.92	0.46	546.68	25.33	70.59	485.13	1.24	1.95

$\overline{\omega}$ = 7.85 ω' = 46.45

CPU UTILITY = 66.91%

DRUM UTILITY = 10.76%

MAGNETIC TAPE UTILITY = 44.34%

NUMBER JOBS PROCESSED = 835

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURA-TION (SEC)	RELAT-IVE RES-PONSE TIME	WAIT TIME (SEC)
0-1	0.27	137.0	48.49	0.76	396.85	0.50	1.45	40.89	18.77	13.64
1-8	3.27	338.0	52.28	0.34	356.95	3.24	8.36	42.87	3.46	9.05
8-120	32.52	447.0	66.39	0.44	470.10	13.04	44.84	100.02	1.57	5.84
120-960	297.57	78.0	66.41	0.46	526.13	34.80	72.67	491.05	1.37	1.35

$\overline{\omega} = 4.55$ $\omega' = 24.33$

CPU UTILITY = 62.68%

DRUM UTILITY = 15.64%

MAGNETIC TAPE UTILITY = 43.27%

NUMBER JOBS PROCESSED = 1,000

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURATION (SEC)	RELATIVE RESPONSE TIME	WAIT TIME (SEC)
0-1	0.28	140.0	48.16	0.64	324.84	0.67	1.39	28.70	21.55	6.48
1-8	3.21	310.0	52.37	0.33	319.03	4.29	8.38	46.61	3.73	12.33
8-120	34.09	372.0	67.12	0.48	578.20	18.37	44.47	116.90	1.55	5.68
120-960	260.94	63.0	67.44	0.51	509.60	41.08	71.22	431.23	1.36	1.16

$\bar{\omega} = 5.46$ $\omega' = 44.37$

CPU UTILITY = 57.37%

DRUM UTILITY = 20.63%

MAGNETIC TAPE UTILITY = 46.93%

NUMBER JOBS PROCESSED = 885

COMPUTE RANGE (SEC)	TIME MEAN (SEC)	NO. OF JOBS	STORAGE BLOCKS (BLKS)	NO. OF TAPES	BLOCKS TRANS. (BLKS)	DRUM DEVICE TIME	PRINTER OUTPUT (BLKS)	DURATION (SEC)	RELATIVE RESPONSE TIME	WAIT TIME (SEC)
0-1	0.32	57.0	47.05	0.77	403.41	1.01	1.44	35.51	5.02	7.45
1-8	2.90	113.0	51.64	0.31	254.60	4.95	8.17	41.41	3.22	10.83
8-120	29.76	143.0	65.27	0.40	419.99	19.78	46.14	109.03	1.65	6.32
120-960	265.73	23.0	67.26	0.43	564.13	52.80	69.57	422.64	1.25	1.61

$\overline{\omega}$ = 2.72 ω' = 7.45

CPU UTILITY = 52.24%

DRUM UTILITY = 22.72%

MAGNETIC TAPE UTILITY = 38.34%

NUMBER JOBS PROCESSED = 336

APPENDIX G

COMPARISON OF THE MEAN DURATION OF SIMULATION RESULTS

The mean duration of a job, $\overline{\Delta T}_j$, is defined to be the mean interval of time elapsed, for the n_j jobs of compute range j, from end of assembly to job completion. This statistic is output by the simulation model (Appendix F) and is given in the Atlas statistics (Table 3.1).

The variation in the mean duration is illustrated (Figure G.1) for each of the four compute ranges at each value of C_1. The mean durations obtained from the simulation run at $C_1 = 1$ for each of the compute ranges are 35.51, 41.41, 109.03, and 422.64. The corresponding values observed in the Atlas statistics are 80.8, 37.7, 102.2 and 456.9. For three compute ranges (j = 2,3 and 4), agreement is good as the deviation is less than 11%. All four values obtained for the mean duration from the simulation model at $C_1 = 1$ compare favourably as shown in Table G.1.

TABLE G.1

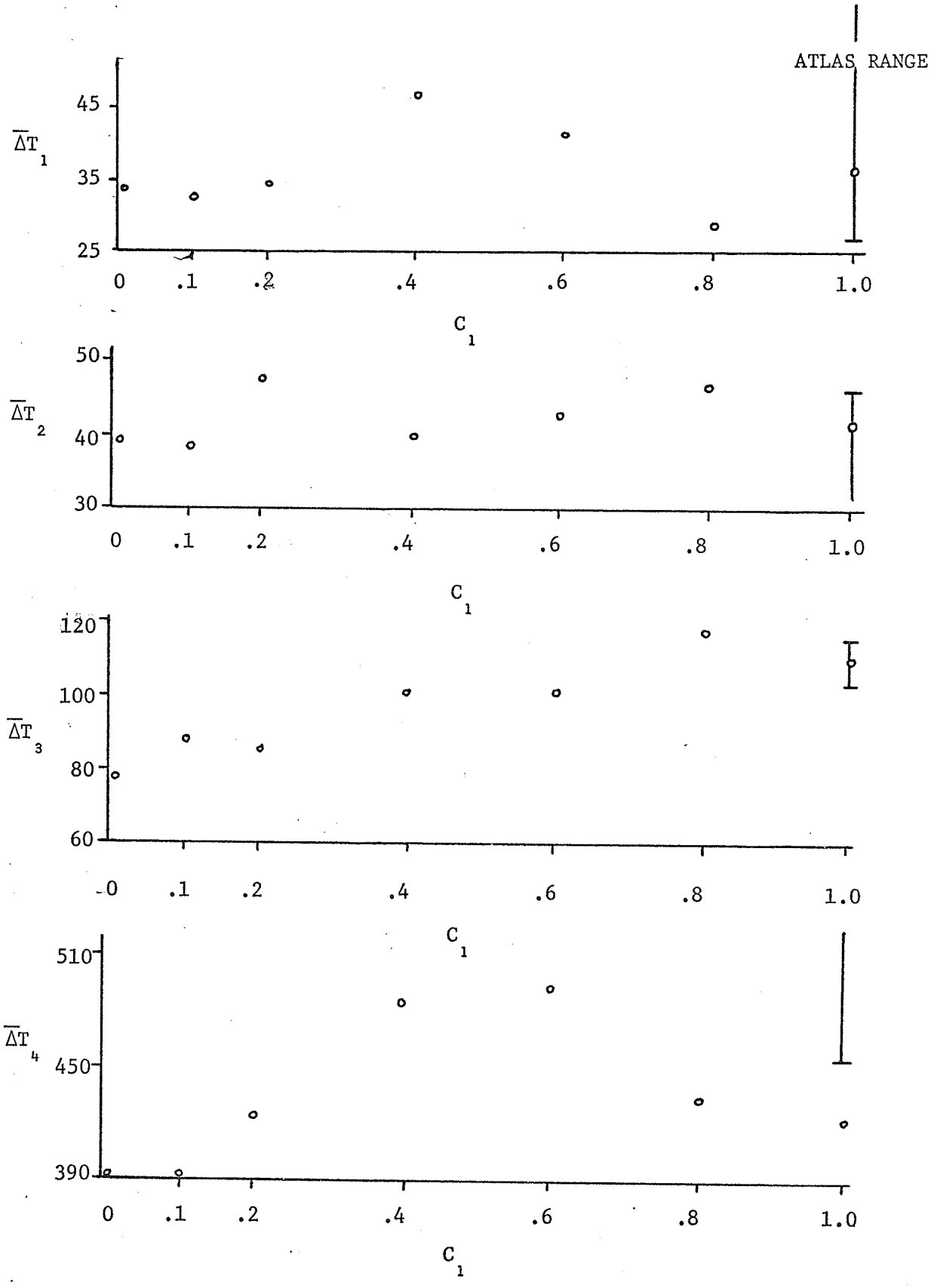
COMPARISON OF $\overline{\Delta T}_j$ TO ATLAS RANGE

COMPUTE RANGE	SIMULATION $\overline{\Delta T}_j$	ATLAS RANGE OF $\overline{\Delta T}_j$
1	35.51	26.1 - 80.8
2	41.41	30.0 - 45.9
3	109.03	102.1 - 113.1
4	422.64	456.9 - 569.4

F I G U R E G . 1

SCATTER DIAGRAM OF $\overline{\Delta T}_j$ VERSUS C_1

o simulation results



APPENDIX H

INTERARRIVAL TIMES GENERATED FROM A DEMAND FUNCTION

In the next event type simulation model developed in this research, events are generated prior to the time at which they occur. This applies also to requests of out-of-core page generated by the model used in Chapter IV involving the demand function, $f_j(p)$.

During the interval of simulated time defined in the model by the time elapsed between the generation of the next out-of-core page request and the actual time this request occurs, the proportion of in-core pages may change. In such an event a job has already received some CPU attention with the initial proportion, p_0, in core storage.

Assume an interarrival time ε_0^* is generated from the distribution $f_j(p)$ with a mean $f_j(p_0)$ at time when the proportion of pages in core for a particular job is p_0. Assume further that the job executes for time $t_1 - t_0$ with proportion p_0 after which time the proportion changes to p_1. The new request time may be defined by the equation:

$$t_1 + \varepsilon_1^* = t_1 + \frac{\varepsilon_0^*}{f_j(p_0)} f(p_1)\{1 - \frac{t_1 - t_0}{\varepsilon_0^*}\} \quad , \qquad .....(H.1)$$

provided that the standard deviation of the distribution is proportional to $f_j(p)$.

If the proporiton changes again to p_2 after $t_2 - t_1$ more units of CPU time, the new request time is given by:

$$t_2 + \varepsilon_2^* = t_2 + \frac{\varepsilon_0^*}{f_j(p_0)} f(p_2)\{1 - \frac{t_2 - t_1}{\varepsilon_1^*}\} \quad . \qquad .....(H.2)$$

The interarrival time ε_i^* can therefore be defined by the recursive relationship:

$$\varepsilon_i^* = \frac{\varepsilon_0^*}{f_j(p_0)} f_j(p_i)\{1 - \frac{t_i - t_{i-1}}{\varepsilon_{i-1}^*}\} \quad , \qquad .....(H.3)$$

for $i \geq 1$ and where ε_i^* is the execution time required to give rise to the next page request after the proportion of in-core pages has changed for the i^{th} time. In order for the requests not to have already occurred, the ratio $(t_i - t_{i-1}) / \varepsilon_{i-1}^* < 1$. The variable ε_0^* is the interarrival time initially generated from the distribution with mean $f_j(p_0)$.

REFERENCES

1. Penny, J.P., "An Analysis Both Theoretical and by Simulation of a Time-Shared Computer System", The Computer Journal, Vol. 9, No. 1, (May, 1966) pp. 53-59.

2. Scherr, A.L., "An Analysis of Time-Shared Computer Systems", MAC-TR-18, M.I.T. Project MAC, Cambridge, Mass. (1965).

3. Nielsen, N., "The Simulation of Time Sharing Systems", Comm.ACM, Vol. 10, No. 7, (July 1967) pp. 397-412.

4. Fishman, G.S., and Kiviat, P.J., "The Analysis of Simulation Generated Time Series", Management Science, Vol. 13, No. 7, (1967) pp. 525-557.

5. Fishman, G.S., and Kiviat, P.J., "The Statistics of Discrete-event Simulation" Simulation Vol. 10, No. 4, (April, 1968), pp. 185-195.

6. Wren, J.M., "Study of Reproducibility of Simulation Studies and Applicability of Cost Curves in Scheduling", M.Sc. Thesis, University of Manitoba, (1969).

7. Abate, J.; Dubner, H.; Weinburg, S.B., "Queuing Analysis of the IBM 2314 Disk Storage Facility", J. ACM (October 1968) pp. 577-589.

8. Chai, L.A., "Study of the Performance of a Scheduling Algorithm for a Time-Slicing Supervisor", M.Sc. Thesis University of Manitoba (1968).

9. McDonald, B.H., "Study of Resource Allocation in Computer Systems by Simulation", M.Sc. Thesis, University of Manitoba (1969).

10. Morris, D.; Sumner, F.H.; Wyld, M.T., "An Appraisal of the Atlas Supervisor", Proc. 22nd ACM. Nat. Conference 1967, pp. 67.75.

11. Howarth, D.J.; Jones, P.D., Wyld, M.T., "Experience with the Atlas Scheduling System", Proc. SJCC, Vol. 23, (1963) pp. 59-67.

12. Kilburn, T.; Payne, R.B.,; Howarth, D.J.,"The Atlas Supervisor", Proc. AFIPS 1961, Eastern Joint Computer Conf., Vol. 20 pp. 279-294.

13. Riding, G., Private communications, Manchester University.

14. Knuth, D.E., The Art of Programming: Vol. II "Seminumerical Algorithms", Addison-Wesley, Reading, Mass. (1968).

15. Fotheringham, J.A., "Dynamic Storage Allocation in the Atlas Computer", Comm. ACM., Vol. 4, No. 10, (October, 1961) pp. 435-436.

16. Kilburn, T.; Howarth, D.J.; Payne, R.B.; Sumner, F.H., "The Manchester University Atlas Operating System, Part I Internal Organization", The Computer Journal, Vol. 4, (1961), pp. 222-225.

17. Kilburn, T.; Edwards, D.B.G.; Lanigan, M.J.; Sumner, F.H., "One-Level Storage", IRE Transactions on Electronic Computers, Vol. 11, No. 2, (April, 1962) pp. 223-235.

18. International Computers and Tabulators Limited, "The Atlas I Computer System Operator's Manual Part One, Central Machine and Supervisor", CS 411, (August, 1964).

19. Fife, D., "An Optimization Model for Time-Sharing", AFIPS Conference Proceedings, Vol. 28, (1966) pp. 97-104.

20. Belady, L.A. and Kuehner, C.J., "Dynamic Space-Sharing in Computer Systems", Comm. ACM., Vol. 12, No. 5, (May, 1969) pp. 282-288.